Application of deep learning methods in materials microscopy for the quality assessment of lithium-ion batteries and sintered NdFeB magnets

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des

Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M.Eng. Olatomiwa, Badmos

Tag der mündlichen Prüfung:	18.01.2023
Hauptreferent:	Prof. Dr. rer. nat. Gerhard Schneider
Korreferent:	T.T.-Prof. Dr. Pascal Friederich

“Your assumptions are your windows on the world. Scrub them off every once in a while, or the light won't come in.”

- Isaac Asimov

Contents

Abstract

Quality control focuses on identifying defects in products and monitoring activities to verify that products meet the desired quality standard. Many approaches for quality inspection use specialized image processing software based on manually engineered features developed by domain experts to recognise objects and analyse images. However, these models are laborious, costly to develop, and difficult to maintain, while the produced solution is often brittle, requiring significant adjustments for slightly different use cases. For these reasons, it is still common for quality inspection in industries to be done manually, which is time-consuming and prone to various human errors. As a result, we propose a more general data-driven approach based on recent advances in computer vision technologies using convolutional neural networks to learn representative features directly from the data. While traditional methods use handcrafted features to recognise individual objects, deep learning approaches learn generalizable features directly from the training samples to recognise different objects.

This dissertation develops models and techniques for the automated detection of defects in light optical microscopy images from materialographically prepared sections. We develop models for defect detection, which can be broadly categorised into supervised and unsupervised deep learning techniques. In particular, various supervised deep learning models are developed to detect defects in the microstructure of lithium-ion batteries, from binary classification models based on a sliding window approach using limited training data to complex defect detection and localization models based on one-stage and two-stage detectors. Our final model can detect and localize multiple classes of defects in large microscopy images with high accuracy in near real-time.

However, successfully training supervised deep learning models typically requires a sufficiently large set of labelled training examples that are often not readily available and can be very costly to acquire. Therefore, we propose two approaches based on unsupervised deep learning for anomaly detection in the microstructure of sintered NdFeB magnets without the need for any labelled training data. The models are able to detect defects by learning from the training data indicative features of only “normal” microstructure patterns. We show experimental results of the proposed defect detection systems by performing quality evaluation on commercial samples of both lithium-ion batteries and sintered NdFeB magnets.

Acknowledgements

This dissertation would not have been possible without the support of many people, and I am grateful for all the amazing people I had the chance to meet during this journey. First, I would like to thank my PhD adviser, Prof. Dr. Gerhard Schneider, for his guidance, support, and constructive feedback during our meetings.

I would also like to thank Dr. Timo Bernthaler for his generous assistance and for fostering a productive research environment at Institut für Materialforschung Aalen (IMFAA).

I also want to thank my colleagues at IMFAA, with whom I have had the pleasure of working. In particular, I am grateful to my research team members Andreas Kopp, Jan Niedermeier, Dominic Hohs, Andreas Jansche, Amit Choudhary, and Florian Trier for the regular group meetings, stimulating discussions, and numerous exchange of ideas.

I would also like to extend my appreciation to all the other members of IMFAA for their warm welcome into the team when I joined and for the pleasure of interacting with and learning from them during my degree.

I would like to express my gratitude to the German Federal Ministry of Education and Research for their support within the program “FH-Impuls” (Project SmartPro, Subproject LiMaProMet, Grant No. 13FH4I02IA) for funding during my PhD.

Lastly, special thanks to my parents, siblings and partner for their endless support and encouragement during my studies.

List of Figures

List of Tables

Abbreviations

General abbreviation

e.g.	**e**xemplum **g**ratia (*en*: for example)
i.e.	**i**d **e**st (*en*: that is)
etc.	**et c**etera (*en*: and the rest)
et al.	**et al**ia (*en*: and others)

Probability and statistics

PDF	**P**robability **D**ensity **F**unction
KL	**K**ullback-**L**eibler
MLE	**M**aximum **L**ikelihood **E**stimation
PCA	**P**rincipal **C**omponent **A**nalysis
ICA	**I**ndependent **C**omponent **A**nalysis

Machine learning

CV	**C**omputer **V**ision
ML	**M**achine **L**earning
SGD	**S**tochastic **G**radient **D**escent
i.i.d.	**i**ndependent and **i**dentically **d**istributed
MSE	**M**ean **S**quared **E**rror
SVM	**S**upport **V**ector **M**achine

Neural network

ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
MLP	**M**ulti-**L**ayer **P**erceptron
GAN	**G**enerative **A**dversarial **N**etwork
NN	**N**eural **N**etwork
ReLU	**R**ectified **L**inear **U**nit
VAE	**V**ariational **A**uto**e**ncoder
R-CNN	**R**egion-**C**onvolutional **N**eural **N**etwork

Lithium-ion battery

Li-ion	**Li**thium-**ion**
LIB	**L**ithium-**I**on **B**attery
NMC	**N**ickel **M**anganese **C**obolt
EV	**E**lectric **V**ehicle
QC	**Q**uality **C**ontrol
SOC	**S**tate **O**f **C**harge

Sintered magnet

HRE	**H**eavy **R**are-earth **E**lements
PM	**P**ermanent **M**agnet
SLM	**S**elective **L**aser **M**elting

Notation

Probability and statistics

σ	Standard deviation
σ^2	Variance
$\mathcal{N}$	Gaussian distribution
E	Expectation
$\hat{p}$	Empirical distribution function

Machine learning

W	Weight matrix
X	Set of training examples
Y	Set of training labels
θ	Model parameters
$\mathcal{L}$	Loss function
η	Learning rate

Chapter 1

Introduction

1.1 Motivation

Quality assessment is an integral aspect of many manufacturing processes. It generally involves evaluating a process to achieve a defined quality standard. Quality control (QC) and quality assurance (QA) are two closely related aspects of quality assessment. QC mainly focuses on identifying defects in products and monitoring activities to verify that the products meet a certain quality standard. While, QA aims to prevent defects by focusing on the process, techniques, and methods used to conceptualize and design the products. In other words, both QC and QA ensure that the design and manufacture of products and services meet consumer expectations.

Nevertheless, due to increasing competition in the manufacturing market, producers have no choice but to increase their production rate while maintaining strict quality control measures. This is especially true for many safety-critical applications, such as in the medical, aerospace, or automobile industry, where the aim is often to realize 100% quality assurance. However, humans often perform these quality inspection tasks manually, which is repetitive and prone to fatigue-induced error. In addition, the analysis can be very subjective and too time-consuming for fast-moving production lines. Therefore, for manufacturers to remain competitive and meet the growing demand for high-quality products, sophisticated visual inspection systems are becoming essential in many production lines.

Furthermore, acquiring relevant information about product quality prevents shipping inferior or faulty products, which can result in substantial financial loss and reputation damage. However, it can also aid with the continuous improvement of existing processes or help to reveal a regression in the current production process. Provided there are suitable measurement techniques for capturing the relevant data, appropriate algorithms for analyzing these data still need to be developed. The basis of most algorithms for automatic surface inspection reply on manually engineered features, which are commonly statistical and filter-based [1]. Even though expert knowledge often allows for powerful features to be created, the process can be laborious and often brittle, needing modification for each new task. Thus, more general solutions which can automatically adapt to new cases would yield significant time and cost savings.

Recent results of applying deep learning [2] have proven to be transformative for various fields, especially in visual recognition tasks. Many state-of-the-art models for image recognition based on deep convolutional neural networks (CNNs) [3] have been able to achieve (and in some specific cases surpass) human-level performance at distinguishing thousands of visual categories [4]. Unsurprisingly, this progress also extends to many other computer vision tasks such as semantic image segmentation [5], object detection [6], and depth estimation [7]. Together, these advances have facilitated many real-world applications, including facial recognition, automated medical image analysis, perception in robotics, and self-driving cars. Recent examples of successful applications of convolutional neural networks on industrial surface inspection problems include steel strip inspection, aluminium profiles, railway track inspection, fabric inspection, LED chips, and LCD panels [8].

In most cases, the defect detection process involves framing the task as either an image classification, object localization, or image segmentation problem [9]. The task is essentially a binary classification or multiclass classification problem for the image classification approach. The object localisation approach aims to draw a tight-fitting bounding box around each detected defect in the image. While in the image segmentation approach, we aim to classify each image pixel as either a defect or not. However, one major prerequisite for successfully training a CNN model is the availability of a sufficiently large set of training data. In fact, depending on the intra-class (instances within the same class) and inter-class (instances from different classes) variance between non-defective areas and the different types of defects, this could require hundreds or up to several thousand training samples. Often, this is an issue since there is usually an abundance of non-defective samples for many properly optimized processes compared to a minimal amount of defective samples, thus making it very difficult to train a supervised machine learning model. The large imbalance between classes often seriously limits the application of classification techniques for defect detection problems. One way to resolve this issue is by switching the training objective from defect classification to anomaly detection, alleviating the need for defective samples during model development. Moreover, a good anomaly detection algorithm would also detect previously unknown classes of defects, thereby constituting a more general solution for quality inspection [10]. In this thesis, we demonstrate the effectiveness of both approaches for automatically detecting microstructural defects in two energy-related technologies. First, we examine the electrodes of lithium-ion batteries and then the microstructure of sintered NdFeB magnets from light optical microscopy images and present this as a new approach for performing a quality assessment.

Lithium-ion battery (LIB). With the increasing demand for energy-storage devices for mobile and stationary storage applications and the advances in renewable energy technology, highly efficient and more cost-effective energy storage solutions are required. Viable electric automobiles and portable electric devices that can operate for long periods without recharging also require more lightweight and more powerful batteries to be developed. Lithium-ion batteries are the most widely adopted energy storage solution compared to other battery technologies. They combine several important benefits such as high energy density, exhibit no memory effects, have low self-discharge rate, and a relatively cheap production cost [11]. As a result, the demand for high-quality lithium-ion batteries is proliferating. The ability of a battery to resist ageing, maintain a good shelf and usage life, and operate well under a variety of conditions are the direct result of adequate quality control measures. However, despite its many advantages, lithium-ion batteries pose safety concerns due to the flammable electrolyte and oxidizing agents present [12]. Internal defects due to poor manufacturing processes, such as low-quality separators, material contaminants, and improperly arranged constituents, can further aggravate the safety risks. Several reports of failure incidents associated with lithium-ion batteries in various battery-powered systems ranging from smartphones and laptops to hoverboards and electric cars include premature battery drain and ageing, overheating, swelling, fires, and explosions. While large customers, such as car manufacturers, are working in direct partnerships with established battery manufacturers to meet their needs and implement their quality requirements, smaller customers of Li-ion batteries reply on the open market supply. Therefore, quality comparison between different battery cells and battery manufacturers becomes indispensable. In addition, early detection of electrode defects can help battery manufacturers to reduce scrap rates after fabrication and testing.

Issues with current approaches. Various destructive and non-destructive examination techniques are used to assess a battery's quality and investigate failure causes at different stages of a battery's lifetime. Non-destructive examination methods do not affect the battery's performance. Therefore, they are commonly used to study the battery degradation process during its operating lifetime and for manufacturing quality control. Electrochemical impedance spectroscopy (EIS) [13], electrical testing methods such as capacity test [14] and incremental capacity analysis (ICA) [15] are the classic non-destructive examination techniques used to characterize battery performance and investigate possible failure causes. For example, by analysing the internal resistance or the charging and discharging behaviour over several charging cycles to determine the loss of capacity due to cell ageing mechanisms. However, despite their prevalence, these methods only provide a summarized view of the battery's performance; they do not provide detailed knowledge about the cause of possible performance and capacity loss. For instance, localized design or material defects such as metallic protrusions, material contaminants, or non-uniform electrode layer structure, which could eventually lead to battery failure, are impossible to detect using these non-destructive examination techniques [16]. As a result, there is increasing use of non-destructive two-dimensional (2D) X-ray analysis and three-dimensional (3D) computed tomography (CT) imaging methods to inspect the internal structure of batteries visually. Figure 1-1 shows a CT scan of an e-cigarette battery containing foreign particles at different locations. The origination of these foreign particles could occur in many ways, for instance, during the coating and assembly process from contaminations existing in the slurries of the active materials. Alternatively, contaminants can emerge during laser cutting of the electrode foils, introducing burrs and metal particles in the electrode material. The welding spatters from the current collector tabs or nickel plating on the casing can also introduce foreign particles into the electrodes. In any case, such foreign particles can pierce the separator and create a short circuit inside the battery, resulting in complete failure. However, even though X-ray analysis can provide fast imaging of large sample areas the resolution is very often too low to properly visualize detailed microstructural properties, thus making it very difficult to distinguish or even detect various types of defects. On the other hand, local microstructure and elemental compositions of each battery component can be directly observed through destructive examination techniques involving battery disassembly. Then, methods such as optical microscopy can be used to obtain a high resolution overview of the microstructure of the sample, while scanning electron microscopy (SEM) and energy dispersive X-ray spectroscopy (EDX) can be used to verify the elemental composition of the active material and to detect the presence of additional phases [17].

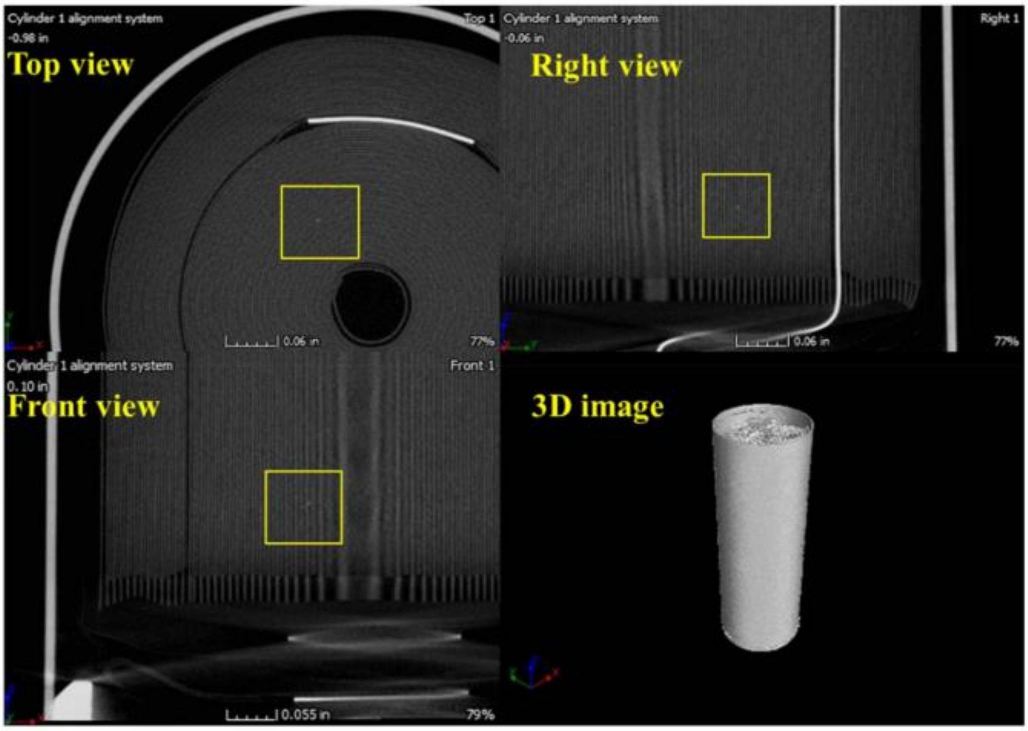


Figure 1-1: CT scan of battery sample with foreign particles (bright points) at different places inside [16].

Sintered neodymium-iron-boron (NdFeB) magnets. Due to their excellent hard magnetic properties, sintered NdFeB magnets have diverse applications, including electric motors, cordless drills, generators, sensors, and hard disc drives [18]. One fundamental property of permanent magnets is their ability to resist demagnetization, known as *coercivity*. However, the thermal degradation of coercivity limits their application in high-temperature environments, for example, in the traction motors of (hybrid) electric vehicles and wind power generators [19]. In order to meet this criterion, higher coercivity NdFeB magnets are required [20]. The two main extrinsic properties of permanent magnets, remanent magnetization M_r and the coervcive field H_c depend on the intrinsic magnetic properties of the main hard magnetic phase (i.e. $Nd_2Fe_{14}B$) and various microstructural features, particularly microstructure defects [18]. The most commonly used approach for developing magnets which can operate in high-temperature environments is by increasing the magnetocrystalline anisotropy of the $Nd_2Fe_{14}B$ phase through direct alloying with heavy rare-earth element dysprosium (Dy) thereby, replacing a fraction (~10 at.%) of the Nd atoms with Dy [19]. However, due to the antiferromagnetic coupling between Fe and Dy this inevitably leads to a reduction in the remanent magnetization M_r and, consequently, in the maximum energy product $(BH)_{max}$. In addition, Dy is an expensive and scarce natural resource, most of which is mined in China, therefore, direct alloying to produce high-coercivity magnets results in very high material cost. Hence, producing high coercivity NdFeB magnets with less or no Dy content is strongly desired.

Nevertheless, since the coercivity of NdFeB permanent magnets is to a large extent determined by the microstructure, especially the grain boundary structure and chemistry, which in turn is determined by the production process, a better understanding of the relationship between the microstructure and coercivity is essential for developing higher-coercivity magnets [20]. Microstructural factors which affect the hard magnetic properties include the mean and standard deviation of the grain size distribution, the orientation degree of the grains, the distribution of the Nd-rich phase, and the residual amount of secondary phases (e.g. α-$Fe, NdFe_4B_4,$ and other borides) present [21]. All kinds of defects, especially soft magnetic phases present in the microstructure, can easily deteriorate the hard magnet properties. Soft magnetic phases provide an easy starting point for a reverse domain, resulting in lower coercivity. Other physical defects, such as irregularly shaped grains and phase boundaries, are also potential sites for demagnetizing fields [19]. Therefore, by eliminating these sites, coercivity can be significantly increased. Alternatively, a quantitative analysis of these sites can aid in comparing the quality of different magnets.

Proposed solution. With materialographically prepared sections combined with microscopy methods, one can obtain information about the frequency and distribution of individual defects from high-resolution images. However, it is still standard practice for materials scientists to visually evaluate the acquired images, where it is easy to overlook inconspicuous faults. Therefore, motivated by the recent achievements of various deep learning methods in computer vision, this dissertation aims to develop deep learning models for automating the detection of microstructural defects from high-resolution microscopy images of lithium-ion batteries and sintered magnets. For instance, given a light microscopy image of a battery's cross-section, the model should accurately identify and categorize all the different types of defects present in the image. It should also provide a concise summary of the defects in an easy-to-view "worst" picture gallery, along with a frequency distribution graph.

Challenges of this approach. In the case of lithium-ion batteries, one common criticism is that destructive methods are not particularly suitable for quality control assessments related to battery fabrication and production because the analysed battery sample cannot be used afterwards [18]. Nevertheless, they are helpful for control measures accompanying the production process. The disassembly of the battery can alter the original failure cause or introduce new types of defects into the electrodes. Hence this approach is more often used in post-mortem analysis of cell components and for investigating the effects of the ageing mechanism on the electrode materials [17], [22]. Due to the highly delicate structure of the battery components, achieving cross sections free of preparation artefacts or the possibility of easily differentiating genuine manufacturing defects from preparation artefacts is another major challenge. In addition, extrapolating the distribution of defects across the entire sample from analysing only a few cross-sections requires some effort to obtain reasonable accuracy. Lastly, compared to many academic research studies that use already prepared datasets, a considerable amount of time and effort is required to collect and label sufficient examples for training a supervised deep learning model. In this case, there is often an abundance of non-defective samples compared to a minimal number of defective samples.

1.2 Contributions and outline

This dissertation develops deep learning computer vision models for detecting microstructural defects from light microscopy images. The outline for the rest of the dissertation is as follows.

Chapter 2 provides relevant fundamentals of lithium-ion battery and sintered NdFeB magnet, describing different types of defects found in both technologies and general techniques for defect detection in image data. Subsequently, we provide relevant mathematical background for neural networks, backpropagation, and various optimization methods. Then a description of the various building blocks of deep learning models for processing images using convolutional neural networks.

Chapter 3 describes the experimental procedure for obtaining high-resolution images from raw samples. It describes the metallographic preparation process for Li-ion battery and sintered magnet, image acquisition, and the tools and software packages used. Subsequently, we develop a deep learning model for defect detection in Li-ion batteries based on image classification to establish a baseline and compare the performance to commonly used traditional approaches based on hand-crafted features. However, an image classification model is typically unsuitable for detection and quantification tasks because it is usually limited to detecting only one object in a given frame, thus resulting in a lower estimation. In addition, since the object can be located anywhere in the frame, it is more difficult to perform correlative analysis to verify the composition of any foreign inclusion. Thus, to specifically address this problem, we develop object detection models based on state-of-the-art architectures, which predict tight-fitting bounding boxes around each detected object along with a category and confidence score.

For detecting defects in sintered magnets, we developed two unsupervised models. The first approach is based on a variational autoencoder network, and the second is based on a conditional generative adversarial network for performing image-to-image translation. The model takes the edge map of the magnet micrograph as input and learns to reconstruct the original image. Since the model is trained on only images without any defects, we can determine the images the model finds more challenging to reconstruct by comparing the reconstruction error between the original images and the reconstructed versions, which would indicate the presence of some anomaly.

Chapter 4 presents the results of the proposed defect detection systems by performing quality evaluation on commercial samples of both Li-ion battery and sintered NdFeB magnets. Specifically, we perform a quality assessment of 18650 round cells and compare three different production batches from the same battery manufacturer. Then, we compare two samples of magnets, one with high magnetic coercivity to one with low magnetic coercivity, produced by different manufacturers. The results are presented using various visualization techniques such as the spatial distribution of the detected regions to get a big picture overview and individually cropped images of defects in a “worst” picture gallery for more detailed analysis. For the unsupervised approach, a heat map visualization of the entire cross-section is calculated based on the anomaly score of each patch.

Chapter 5 discusses how the proposed models perform compared to manual human inspection and traditional approaches. Then we discuss the relevance, benefits, and advantages of the proposed methods. We assess the various models in terms of their speed-accuracy trade-offs, and finally, we discuss the limitations of the proposed methods.

In **chapter 6**, we make overall conclusions, discuss the application of this technology, identify the remaining challenges, discuss some extensions and other applications of the proposed methods, and suggest directions for future work.

Chapter 2

Fundamentals

2.1 Relevant fundamentals of lithium-ion battery

A pressing concern of the present fossil fuel energy economy is the associated carbon dioxide (CO_2) emissions which have increased at a constant rate, with a remarkable increase over the last 30 years [23]. From 1970 to 2005, the CO_2 level has almost doubled, resulting in a global rise in temperature with associated series of effects due to climate changes, whose impacts are expected to intensify in the coming decades if significant changes are not made. As a result, the continuous need for energy renewal requires a much higher level of clean energy sources than what is presently in force. Amidst stricter emission rules and governments offering generous incentives, electric vehicle (EV) sales have accelerated rapidly in recent years. The increasing market penetration of EVs has led to enormous demand for energy storage systems with improved performance, longevity, and reliability. Accordingly, investments in the exploitations of renewable energy sources are increasing worldwide, with particular attention to wind and solar power energy plants.

Moreover, the intermittence of these resources also requires highly efficient energy storage solutions. Electrochemical systems, such as batteries and supercapacitors, can efficiently store and deliver energy on-demand in stand-alone power plants and provide good power quality and load levelling of the electrical grid in integrated systems. These systems are already playing an important role in the field, with major benefits exhibited for various renewable energy projects [23], [24]. Nevertheless, the effectiveness of batteries in these applications is directly related to their energy content and lifetime, of which there are various types, such as lithium-ion, sodium-ion, lead-acid, nickel-metal hydride, zinc-air batteries etc. Of these technologies, lithium-ion batteries have become ubiquitous due to an unmatched combination of high energy and power density, relatively long cycling life, and portability, which makes it the technology of choice for many portable electronics, power tools, and electric vehicles [25].

Regarding sustainable power supply, Li-ion batteries also play an important role in stationary storage of energy produced by renewable sources. The high energy efficiency of Li-ion batteries allows their use in various electric grid applications. This includes improving the quality of energy harvested from wind, solar, geothermal, and other renewable sources and contributing toward the wider adoption for building an energy-sustainable economy [11]. As a result, Li-ion batteries are of major interest to industry and government funding agencies, whereby research in this field has thrived in recent years. The goal of current energy storage device developments can be grouped into five categories [26]: (1) lowering the cost, (2) improving performance and efficiency, (3) promoting reliability and durability, (4) ensuring usage safety, (5) reducing the environmental impact.

In certain applications such as electromobility and electric grid, Li-ion batteries are currently costly, which is a major factor inhibiting its widespread adoption. A shortage of lithium and some other transition metals currently used in Li-ion batteries may also become an issue one day. At present, lithium is not a major factor in the cost of Li-ion batteries. The use of lithium is mainly in the cathode and electrolyte, which makes up only a small fraction of the overall cost [11]. The majority of the cost can be attributed to two main contributing factors, the cost of processing the components and the cost of cobalt used in the cathode. Nevertheless, Li-ion batteries have certain fundamental advantages over other chemistries. Lithium has the lowest reduction potential of any element, allowing batteries based on Li to have the highest possible cell potential. In addition, Li is the third lightest element and possesses one of the smallest ionic radii of any single charge ion, enabling Li-based batteries to have high gravimetric and volumetric capacity and power density [11]. For these reasons, Li-ion batteries will most likely continue to dominate the portable electrochemical energy storage space for many years to come. Therefore, improving the cost and performance will greatly expand their applications and facilitate new technologies which depend on energy storage.

Much of the research in Li-ion batteries focuses on the electrode (anode and cathode) materials. Research on electrodes with higher rate capability, higher charge capacity, and cathodes with sufficiently high voltage can improve the energy and power densities of Li-ion batteries and make them smaller and cheaper. However, a more pressing development objective is to enhance cell quality and service life since requirements in the new fields of application are considerably more stringent in terms of higher cost and safety standards. In addition, the operating conditions are more demanding than with existing ones [27]. The cycle life of a battery is governed by a wide range of factors [28], [29], which include the temperatures the cells are exposed to [30], rate of operation [31] and the depth of discharge of each cycle [32]. However, of utmost significance is the manufacturing quality of the electrodes and the electrode winding (jelly-roll), which essentially determines 80-90% of the cell quality [27].

Moreover, the electrodes are fundamentally defined by the active materials used in the active mass (e.g., chemical composition, crystal structure), the microstructure (e.g., the content of porosity, phase fractions), and the fine geometry (e.g., the thickness of the coating material). Furthermore, various studies have shown that initial defects in the jelly-roll are nucleation points for electrode deformation, which can be correlated to cell ageing and untimely failure [33]–[35]. However, in battery manufacturing, mostly in-line mechanisms are applied to evaluate and ensure the quality, which does not provide any information on the microstructure of the electrodes, which ultimately determines the battery quality. Many varied degradation mechanisms can reduce the performance of a battery, and there are several similarly varied techniques for tracing them, including acoustic time-of-flight measurements [36], electrochemical methods [37], and X-ray computed tomography (CT) among others. X-ray CT is a powerful non-destructive technique for investigating batteries, with various studies highlighting its effectiveness in identifying cell failures [38], [39]. However, the resolution is often not detailed enough to properly visualize various microstructural properties. Therefore, there is still a lack of appropriate characterization tools to facilitate the evaluation of manufacturing processes and aid with further development and improvements of current manufacturing practices.

Structure, function, and cell formats

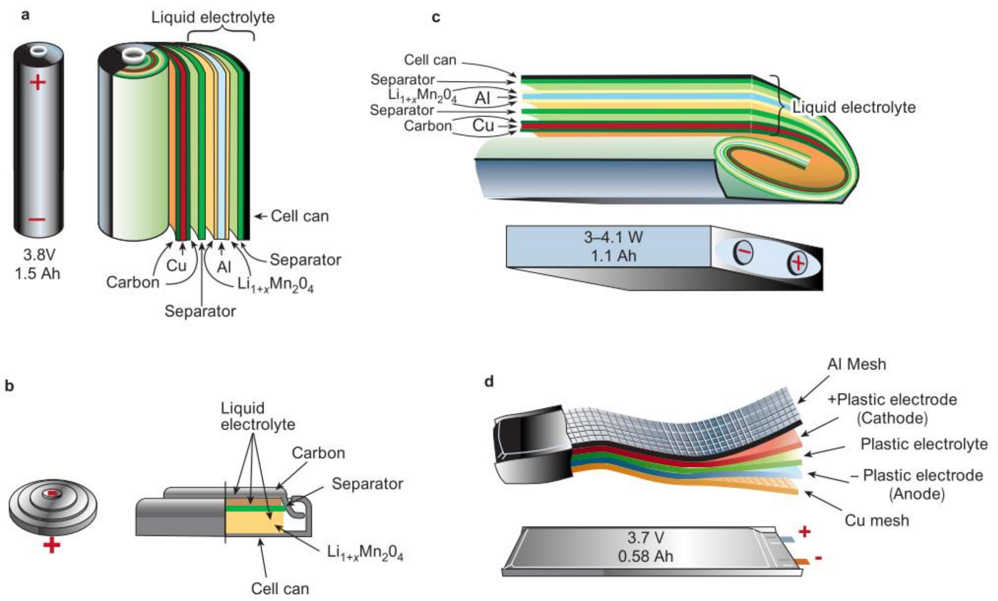


Figure 2-1: Schematic illustration showing the shape and components of various cell formats of lithium-ion batteries: (a) round cell, (b) button cell, (c) prismatic cell, (d) pouch cell [40].

Li-ion cells, which are the basic electrochemical unit containing the electrodes, separator, and electrolyte, are designed and produced according to the specific application and thus are available in various formats, which can generally be divided into four groups: button cell, cylindrical cell, pouch cell and prismatic cell [40]. Li-ion batteries in button cell format are in high demand for applications such as measurement systems, Internet of Things (IoT) sensors, car locking systems, Bluetooth wireless headsets etc. Flat or pouch cells are commonly used in mobile phones, tablets, and power banks. Due to their relatively high gravimetric energy density, this cell format is also increasingly used in electric mobility. Cylindrical or round cells are the most common cell types used in many consumer applications. They are available in various sizes, such as the 18650 format (18 mm diameter x 65 mm height) used in power or gardening tools and e-bikes, with electric vehicles and stationary energy storage systems being the most popular application.

On the other hand, prismatic cells were developed specifically for stationary storage systems and electric mobility. An illustration of these cell formats is shown in Figure 2-1. Due to their hard cases, cylindrical and prismatic cells are considered to be relatively safe and therefore chosen by most automotive OEMs for their batteries. On the other hand, Pouch cells have an arrangement of electrodes and separator layers packed in a compound foil, which usually consists of aluminium and polyolefin layers. The absence of a case gives pouch cells the highest gravimetric energy density. However, for many practical applications, they still require an external means of containment to prevent expansion when their state of charge level becomes high and for general structural stability of the battery pack. This cell type is easy to build in laboratories in small numbers without significant investment in expensive production facilities. The pouch and prismatic cells provide a higher packing density on the battery system level than the cylindrical cells due to geometric constraints in most applications. However, the faster production speed is one of the main advantages of cylindrical cells compared to cells with stacked electrodes.

Like other cell types, a Li-ion battery cell comprises an electrochemically coupled anode and a cathode. For rechargeable cells, the term anode (or negative electrode) refers to the electrode in which oxidation occurs during the discharge cycle, while the other electrode is the cathode (or positive electrode). The positive electrode becomes the anode during the charge cycle, and the negative electrode becomes the cathode. For most Li-ion cells, the lithium-oxide electrode is the positive electrode. The two electrodes are electrically isolated from each other by a separator through which lithium ions can pass; however, it prevents the direct exchange of electrons between the electrodes. In Li-ion cells, thin aluminium and copper foils (current collectors), coated on both sides with a porous active mass, serve as the electrodes. Lithium ions move from the negative electrode through the electrolyte to the positive electrode during discharge and back when charging. At the same time, the current collectors serve to emit and collect electrons, as shown in Figure 2-2. A more detailed description of the process involved during charging and discharging can be found in [23], [40], [41].

The active mass of the anode and cathode mainly consists of the so-called active materials (storage particles), which store the lithium ions cycling between the negative and positive electrodes during charging and discharging in their crystal lattice structure in an intercalation process. Li-ion batteries use an intercalated lithium compound as the active material of the positive electrode. For commercial cells, graphite is often used as the active material of the negative electrode. Lithiated transition metal oxides with strongly differing stoichiometry are often used as the active material of the cathode. Previously, lithium cobalt oxide $LiCoO_2$ (LCO) or lithium manganese oxide $LiMn_2O_4$ (LMO) were commonly used as the cathode active material. However, in the most recent generation of cell chemistries, mixed oxides are often used, in which individual Co atoms in $LiCoO_2$ are substituted by either nickel and or manganese atoms. This formation of lithium nickel manganese cobalt oxide $Li(Ni_xCo_yMn_z)O_{2,}$ commonly referred to as NMC, exhibits some very advantageous properties such as higher energy and power density with enhanced cycling stability compared to cells based on LCO intercalation material. The electrode and cell capacity are determined by the amount of lithium ions that these active materials can store. To ensure the integrity of the storage particles in the cathode coating, in addition to the active material, the active mass coating also comprises a polymer binder and conductive additives. The conductive additives (most often carbon blacks) improve the electrical contact between storage particles and the current collector, which helps to reduce the internal resistance of the cell. The polymer binder (for example, polyvinylidene fluoride PVDF) ensures the integrity (cohesion) of the entire coating and the adhesion of the active mass on the metallic current collector. Electrode coasting is compacted for optimal volumetric energy densities by calender rolls to determine the final coating thickness. This improves the electrical contact of the entire electrode coating and reduces the electrode porosity, which plays a significant role in regards to the quality of the active mass. During operation, the porous space around the storage particles is filled with electrolytes, determining the cell's ionic conductivity and internal resistance.

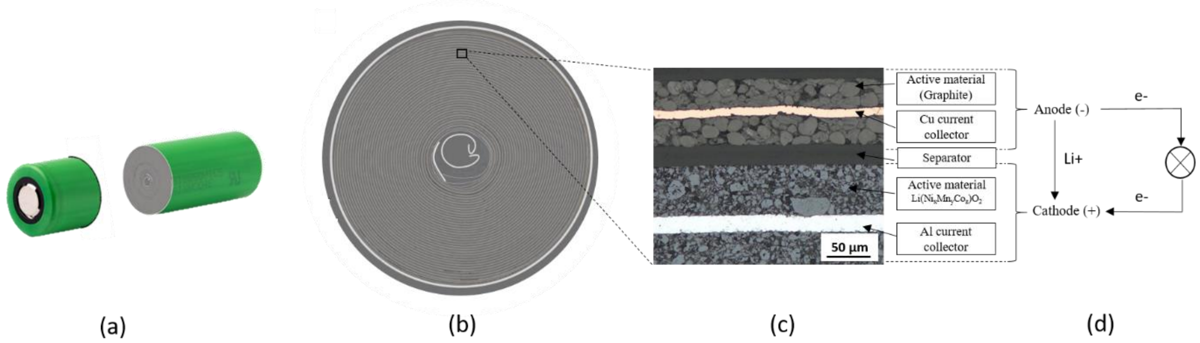


Figure 2-2: (a) Schematic representation of a sectioned cell. (b) Light microscope cross-section overview image. (c) Light microscope image details showing the structure of a 18650 type round cell. (d) Schematic flow of electron and ion during the discharging process.

The manufacturing process of Li-ion cells

First, basic materials such as anode and cathode active materials, electrolytes, additives, and binders are synthesized. The choice and purity of materials significantly determine the electrical and chemical properties of the cell. The active materials in the powder form are mixed with the conductive additives and the binder with a solvent to form the so-called slurry. A homogeneously dispersive slurry is aimed at using mixing tools and then coated onto the respective current collector foils using, for example, the slot-die method [42]. The slurry is coated on aluminium foils for the cathodes and copper foils for the anodes. The foil thickness is typically between 10 and 25 μm. After the coating, the active layer is heated in a controlled manner in order to dry out the solvent used for slurry production and to fix the coating on the foils through the binder. During the drying process, the coated electrodes are subject to multi-stage temperature gradients, individually varied according to the wet layer thickness. The average drying temperatures are between 80°C and 160°C. After the drying process, the porosity of the electrode coating is typically around 50%.

In the next phase, the electrodes are compacted through calendaring. The compression level determines the final active mass volume and other factors that considerably influence the cell’s performance. For instance, the electrical conductivity increases with a higher compaction gradient due to improved contacts between the storage particles, the conductive additives, and the current collector foils. On the other hand, the porosity and thus the ionic conductivity decreases due to poor penetration of electrolytes. Subsequently, the electrode layers are cut into appropriate sizes by either mechanical or laser cutting tools. Three different cell stacking methods exists, which differ in how the electrode and separator materials are processed. Single sheet stacking uses single electrodes and single separator sheets to isolate the anode and cathode, winding technologies process continuous electrode materials for use in cylindrical or prismatic cells. On the other hand, Z-folding processes can use either continuous electrodes and separators or only continuous separators combined with single electrode sheets to form the so-called jelly-roll.

After building the stack, the cell is assembled by joining the current collectors using an ultrasonic or laser welding process. For prismatic or round cells, the cell stack is housed in compound foil or aluminium cases and filled with electrolytes before sealing the enclosure. In the final stage, the cell is charged and discharged for the first time in a process called formation. During this process, lithium is irreversibly deposited on the surface of the graphite structure, forming a boundary layer between the electrolyte and graphite known as the solid electrolyte interface (SEI), which is highly responsible for the operation quality and the cycle stability of the cell. However, since all the manufacturing steps described above can only be performed within a specific processing window, there are various stages in which potential faults may occur in the finished cells due to manufacturing errors or defects that negatively impact the cell quality.

2.1.1 Quality control of lithium-ion battery

Quality control (QC) in the production of Li-ion cells is fundamentally related to safety and cost reduction. Reducing the manufacturing cost of lithium-ion batteries remains a challenge in the industry, and detailed attention to fundamental and applied science is required [34]. Strategies for reducing the manufacturing cost of LIB include implementing low-cost processing, further development of new and existing testing methods, and adequate quality control tools to reduce the scrap rate. In addition to the quality control tools, it is also important to understand the correlation between manufacturing defects and the performance of the electrode in order to establish clear pass/fail criteria.

Types of defects in Li-ion battery and their impact on performance

The mechanism of defect-induced performance degradation in Li-ion batteries is diverse and can be primarily linked to various electrochemical processes and defect evolution. Several types of defects which can be introduced during the electrode manufacturing process are presented in Table 2-1.

Table 2-1: Presents a list of possible defects during the cell manufacturing process [34], [43].

Component	Sub-category	Reason	Worst-case outcome
Electrode coating	Homogeneity	Non-uniform coating can lead to the variability of the local capacity of the electrode.	Reduced cycle life
	Thickness	Thicker layers result in a higher capacity due to the higher amount of storage particles. However, at high discharge rates, they negatively impact the capacity and cycle stability of the cell.	Unstable performance
	Particle sizes	The size of the particles of the active materials is essential for the performance and capacity of the cell.	Insufficient performance

	Phase components	Inhomogeneous mixing of two active materials can lead to uneven phase contents and thus brings the potentials out of balance	Capacity losses, Li deposition
	Blisters/agglomerates	Agglomerates of inactive components such as binders and carbon black can lead to higher content of inactive components and lower gravimetric capacity.	Capacity losses
	Porosity	Highly compressed electrode coating makes it difficult for electrolytes to lubricate the storage particles.	Reduced performance
	Divots/pinholes	Areas with missing coating caused by captured bubbles in the slurry reduce the amount of active materials and can expose the current collector to the electrolyte.	Reduced Capacity
	Cracks/fracture	Fracture of storage particles during calendaring or cracks in the electrode coating during winding can result in reduced connectivity	Insufficient performance, Capacity fading
	Water	Contamination by water can lead to a chemical reaction with the electrolyte resulting in abrasive hydrofluoric acid	Reduced cycle life, HF Corrosion
Foreign particles	-	Foreign particles (especially metal particle contaminants) can penetrate the separator membrane	Internal cell short-circuit
Separator	Thickness	Particles exceeding a size equivalent to the thickness of the separate can penetrate	Internal cell short-circuit
Current collector foil	Collector thickness	Excessively thick collector foils lead to lower gravimetric and volumetric energy density, while the opposite may result in the foils cracking due to stress.	Tearing of the collector foil
	Number of current collectors	Missing current collectors can lead to heat accumulation	Insufficient heat dissipation
	Structure of current collector	Deformation or burrs in the current collector foils due to laser cutting can damage the separator	Internal cell short-circuit
	Electrode contacting	Welding burrs can damage the separator foil	Internal cell short-circuit
	Positioning	Misalignment can lead to the deformation of protruding anode layers	Internal cell short-circuit
Housing	Sealing	Escaping of hazardous substances such as electrolyte	Corrosion, Ecologically critical
	Diameter	Insufficient inner diameter of the cell housing can lead to telescoping of the electrode winding during cell assembly	Internal cell short-circuit
	Electrode contacting	A poor connection can lead to high electrical resistance	Local hotspots

	Height	Insufficient height can lead to incorrect positioning of cell windings and deformation of protruding layers	Internal cell short-circuit
Safety valve	-	Inactivation can lead to the build-up of internal pressures	Explosion
Cell winding	Telescoping	Telescoping can lead to the deformation of protruding anode layers	Internal cell short-circuit

2.1.2 Application of machine learning in battery technology

Given the high number of factors that could influence the overall performance, efficiency, safety, and cost of Li-ion batteries, researchers are increasingly looking to apply machine learning techniques to guide their performance analysis and design choices by distilling useful information from a large amount of measured data. For example, this could be relevant for selecting appropriate structural and processing parameters, influencing the choice of materials, and designing operational strategies. This section will look at various applications of machine learning in the development and management of batteries. The types of problems that machine learning is applied to in this area include state estimation and prediction [44]–[47], property analysis, and classification [48], lifetime prediction [49]–[51], fault detection and diagnosis [52]–[58], together with modelling [59], [60], design [61], [62] and optimization [63].

In the work of [44], they applied Gaussian process regression (GPR) to forecast the state of health (SOH) of batteries. In [64], they applied GPR to estimate the in-situ capacity of Li-ion batteries. The work of [45] applied various GPR models (including regular GPR model, recurrent GPR model, and autoregressive recurrent GPR model) to estimate the state of charge (SOC) of Li-ion batteries. In [46], a random forest regression model was applied to estimate on-line SOH with a mean square error (MSE) below 1.3%. Models based on artificial neural networks (ANN) are also widely used for battery state estimation. In [47], the voltage, temperature, average current, and average voltage of the battery at time t are used to train a deep neural network to estimate the SOC of Li-ion batteries with an average MAE of 1.10% at 25°C and 2.17% at -20°C. In the work of [49], a linear regression model was used to predict the battery cycle life before capacity degradation based on data from the early discharge cycle. In the work of [50], a k-NN regression algorithm was used to estimate the remaining useful life (RUL) of Li-ion batteries by estimating from cells sharing a similar degradation trend. Another notable application of machine learning approaches is detecting battery defects and classifying abnormal batteries. In the work of [52], several machine learning algorithms, including k-NN, kernel-SVM, logistic regression, Gaussian naïve Bayes (GNB) and artificial neural networks, were applied to classify the unbalanced and damaged Ni-MH battery cells into two classes based on 28 discharging voltage curves. In [53], a hierarchical clustering algorithm was applied to time series data to detect defects in lead-acid batteries. Approaches based on deep learning are also commonly used for detecting and classifying batteries with abnormal behaviours. In [54], a deep belief network model was used to detect anomalies in the voltage of storage batteries. A wavelet-neural network system containing discrete wavelet transform (DWT) and general regression neural (GRNN) network were used to detect faults in Li-ion batteries for electric vehicles [55]. The machine learning approaches mentioned above are mainly applied to low dimensional structured data as the input for training the model. In some other application, the input to the machine learning model is in the form of images, such as a high resolution picture of the battery electrode microstructure. In this case a convolutional neural network (CNN) which is capable of extracting relevant features from high dimensional unstructured data is typically applied. For example in the work of [56] a CNN model was used to classify the voltage of fully charged LiPo batteries after a period of usage based on RGB images obtained from signal reading of a Walabat sensor in order to learn images of “battery face”. In the work of [57] various CNN based models were trained to detect microstructural defects such as the presence of foreign particles and layer deformation in the electrodes of Li-ion batteries due to various manufacturing errors in order to evaluate the battery quality. They also compared the performance of CNN based models to those of classical machine learning approaches based on handcrafted features. In a different study [58], several machine learning algorithms including SVM, NN, CNN, and RCNN models were applied to detect blister defect in sheets of polymer Li-ion battery. It was found that the CNN model achieved the best classification performance with an F_1-score of 0.988.

2.2 Relevant fundamentals of sintered NdFeB magnet

Neodymium-iron-boron (NdFeB) magnets have been employed in various applications from sensors, actuators, motors, generators etc., due to their excellent magnetic properties [65]. For one, they are the strongest type of permanent magnets available commercially and provide the best saving in weight per joule of energy supplied [66]. In recent years, there has been an increasing demand for higher coercivity magnets for various reasons, one of which is the increasing demand for electromobility, which requires a transition from fossil fuel-powered combustion engines to electric drive motors. For such applications, there are several ways to develop an electric motor. An electrically excited coil set or a permanent magnet (PM) can generate a magnetic field within the electric motor. Nevertheless, the motor must be able to operate at high temperatures and exhibit high torque while minimizing cost and weight [67].

Permanent magnets are materials that retain a usefully large magnetic moment after being exposed to an external magnetic field. Electric vehicles are considered one of the largest markets for permanent magnet materials. The main benefits of using a brushless permanent magnet in automotive motors include no excitation losses, higher torque per volume, higher magnetic flux density in the air gap, and reduced maintenance [68], [69]. In addition, the material must also have enough coercivity to resist demagnetization. For this reason, neodymium-iron-boron materials (NdFeB) are typically used.

For a permanent magnetic material to succeed technologically, it must have a high Curie temperature. Furthermore, for application in higher temperature environments, materials with increasingly higher values of Curie temperatures are required, such that the magnetic properties must be reasonably stable for long periods in adverse environments. However, pure NdFeB magnets have low operating temperatures ranging from 80 °C to 100 °C, which can be increased by including additives [70]. Many efforts have been carried out to enhance the temperature stability of NdFeB magnets. Direct alloying with heavy rare-earth elements (HRE) like terbium and dysprosium can be used to significantly raise the temperature stability by partially replacing Nd in the 2:14:1 phase lattice to increase the coercivity [18]. For instance, adding 10 wt % of dysprosium can extend the temperature stability up to 200 °C. Likewise, terbium has a greater effect on coercivity than dysprosium but has a higher price and lower availability. However, dysprosium and terbium are expensive and scarce resources such that direct alloying significantly raises the material cost. The largest amount of rare earth containing ores is located in China, which has 75% of the world's reserves. The USA has 13% of the world's reserves followed by India and Australia, which have 4% and 1.5%, respectively [66]. Therefore, producing high coercivity NdFeB magnets with less or no rare-earth content is of high strategic interest. For these reasons, optimal control of the magnet's microstructure becomes increasingly important.

Magnet properties

The tendency for ferromagnetic materials to remember their magnetic history is called hysteresis. Figure 2-3 shows a hypothetical hysteresis loop for a permanent magnet. It is a measure of magnetization (M) versus applied magnetic field (H), which effectively measures the internal response of the magnet to an externally applied field. Hysteresis curve also measures magnetic flux density (B) versus applied magnetic field (H). The following equation describes the relationship between magnetic flux density and magnetization:

$$B = \mu_0(H + 4\pi M) \qquad (2.1)$$

Some useful properties for characterizing a magnetic material can be obtained from the hysteresis behaviour. This includes:

Saturation Magnetization (MS). It is the maximum magnetization that a material possesses once fully magnetized. It is denoted by the maximum magnetization value on a hysteresis curve.

Remanent Magnetization (MR). After magnetization with an externally applied field, the magnetization that persists when the external field is removed is called the remanence. This is represented by the point where the hysteresis curve crosses the positive vertical axis.

Intrinsic Coercivity (Hci). This is the opposing magnetic field required to completely demagnetize the material. It is signified by the point where the M versus H curve crosses the negative horizontal axis.

Technical Coercivity (Hc). Denoted by the point where the B versus H curve crosses the negative horizontal axis. It is the opposing magnetic field needed to reduce the magnetic flux density to zero.

Curie Temperature (Tc). The temperature at which the material no longer displays long-range ordering or hysteretic magnetic properties due to thermal agitation.

Maximum Energy Product (BHmax). Is a measure of the maximum potential energy that can be stored within the magnet. It is determined from the demagnetization behaviour of the material.

These properties arise from several sources that can be either intrinsic or extrinsic. Some properties are derived from the composition and the crystal structure of the material, while others are based on a combination of the composition and crystal structure as well as the microstructure of the material. Properties that occur independently of the microstructure are known as intrinsic properties. Magnetocrystalline anisotropy, saturation magnetization and Curie temperature are examples of intrinsic properties. In contrast, extrinsic properties are those that are dependent on the microstructure or other external factors. Remanent magnetization and energy product are two such examples. Coercivity depends on intrinsic factors such as magnetocrystalline anisotropy and is also greatly affected by the microstructure.

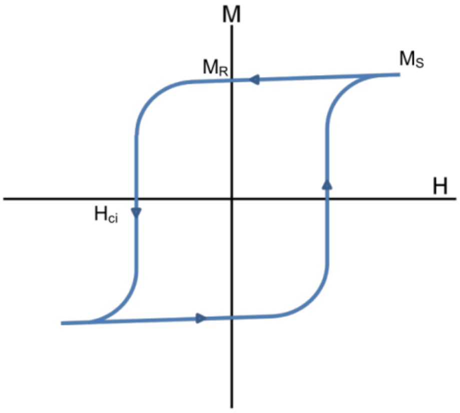


Figure 2-3: Example model of magnetization M against magnetic field H [67].

A NdFeB magnet is the most widely used type of rare-earth magnet. It is made from an alloy of neodymium, iron and boron to form the $Nd_2Fe_{14}B$ tetragonal crystalline structure. Since its discovery in 1984, it has developed rapidly into a technologically important material and replaced other magnet types in many applications that require strong permanent magnets. In pure form, neodymium is also a ferromagnetic material, which means it can be magnetized to become a magnet, but its Curie temperature is extremely low. However, compounds of neodymium with transition metals such as iron can have a much higher value of Curie temperature and thus are used to make neodymium magnets. The cost and abundance of the starting materials, in addition to the intrinsic properties, are what make NdFeB magnets technologically successful. The 2:14:1 phase consists of approximately 72 wt % Fe, which is very cheap and nonstrategic, while neodymium, a rare-earth element, is still more abundant than many common metals such as tin, lead, and silver. On the other hand, boron is expensive but is only required in small amounts. The strength of neodymium magnets is due to several contributing factors, of which the most important is the tetragonal $Nd_2Fe_{14}B$ crystal structure which possesses remarkably high uniaxial magnetocrystalline anisotropy [71]. This basically means that the material's crystal magnetizes along a specific crystal axis and is very difficult to magnetize in other directions. The magnet alloy is composed of microcrystalline grains aligned in a strong magnetic field during production, so their magnetic axes all point in the same direction.

Depending on the manufacturing process, significant differences can exist in the microstructure found in NdFeB magnets produced by sintering or melt-spinning. In sintered magnets, 80 to 85% of the magnet comprises the 2:14:1 phase, while in melt-spun magnets, over 95% of the volume is the 2:14:1 phase. The ideal microstructure for sintered magnets would be such that the nucleation of reverse domains is inhibited. The optimum grain sizes in sintered magnets are determined to be between 5 and 10 μm. Defects are potential sites for the nucleation of reverse domains and the reduction of energy product. In addition, a nonmagnetic rare-earth eutectic phase (grain boundary phase) should surround each grain to magnetically insulate each grain to prevent the nucleation of one grain from spreading to neighbouring grains. Figure 2-4 shows the microstructure of a NdFeB magnet along with the different phases.

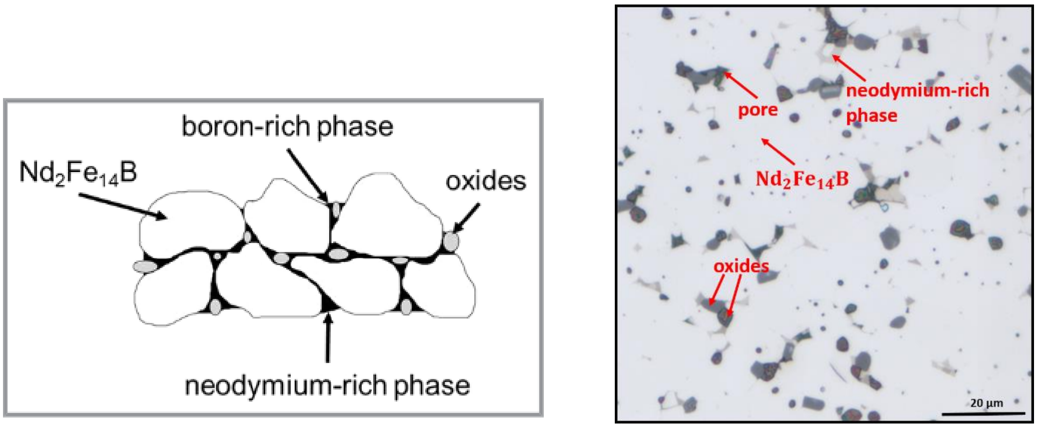


Figure 2-4: (Left) Schematic illustration of NdFeB magnet microstructure [72]. (Right) Actual microstructure of NdFeB magnet acquired using a light optical microscope. It is characterized by grains on the order of 10 μm or less, surrounded by a thin layer of a secondary rare-earth phase.

The manufacturing process of sintered NdFeB magnets

The conventional production process for sintered NdFeB magnets generally consists of the following steps: vacuum induction melting, crushing, milling, pressing and sintering of the magnet material. Nd-Fe and Fe-B master alloys are used as the input material. In the first step, the alloys are subjected to vacuum induction melting. Then the ingot obtained from the melting of the alloys is crushed and milled to < 5 μm sized powders before the monocrystals can be aligned in a magnetic field of approximately 1 Tesla [66]. The aligned powder is then cold-pressed using pressures on the order of 200 MPa to approximately 70% density to ensure that there is enough contact between particles to prevent rotation. The compacted material is then sintered at approximately 1100 °C in an argon-protected atmosphere to full density. Usually, a lower temperature post heat treatment in the 700 °C to 900 °C range is necessary to achieve proper grain boundary phases. In the next step, the material is machined into its final shape, and finally, the material is magnetized [70]. The magnet can either be coated with organic resin or metallic coatings such as Al or Zn to increase the resistance to corrosion. The final microstructure is characterized by grains on the order of 5 to 10 μm, surrounded by a thin layer of a secondary rare-earth rich phase.

2.2.1 Quality control of sintered NdFeB magnets

As described in the previous section, coercivity is an integral property of a permanent magnet since, without substantial coercivity, the magnet would be easy to demagnetize and therefore would not be very useful in many applications. A magnetic material must have anisotropy in order to have coercivity. Magnetocrystalline anisotropy comes from the energy difference between the easy and hard magnetization directions. In materials containing rare-earth elements, the magnetocrystalline anisotropy can be very large. Therefore, demagnetization typically occurs through domain processes which can be either domain nucleation or domain growth. Domain nucleation typically occurs first in areas that contain chemical or physical defects. Chemical defects such as soft magnetic phases with lower magnetocrystalline anisotropy provide an easy starting point for a reverse domain, leading to low coercivity. At the same time, physical defects typically from surface irregularities, abnormally shaped grains, dislocation or phase boundaries are also sites for demagnetizing fields. Therefore, coercivity can be increased in material controlled by domain nucleation by eliminating or minimizing the number of defect sites that reverse domains nucleate. A single domain particle can be formed if the grain size is significantly reduced. In addition, the presence of secondary phases at grain boundaries can help to isolate the particles and smooth the grain boundaries. This is often used in NdFeB type magnets. Another approach for improving coercivity is through domain wall pining. Once domains are nucleated, they can be prevented from growing by incorporating defects into the bulk of the material. These defects are often secondary particles that can interact with the domain walls, reducing the total wall energy. The most effective defects must be sized within the same order of magnitude of the domain wall thickness and dispersed throughout the structure. The distribution and size of the pinning centres and the flexibility of the domain walls will therefore determine the coercivity in pinning controlled magnets. In general, the main areas for microstructural improvements in sintered magnets focus on reducing the number of defects, grain size, and magnetic coupling between the grains [73]. Figure 2-5 shows some examples of defects found in NdFeB type magnets.

2.2.2 Application of machine learning in rare-earth magnets

This section outlines various applications of machine learning in the characterisation, design, and development of magnets. In [74], a regression model based on XGBoost [75] and an evolutionary algorithm were used to accurately predict magnetic characteristics, such as iron loss and permeability, in order to suggest optimal processing parameters which can be used for selective laser melting (SLM) based on the given magnetic characteristics. For both targets, iron loss and permeability, they achieved R^2 scores above 0.9. An approach that uses composition to estimate coercivity and maximum energy product in (PrNdLa-Ce)$_2$Fe$_{14}$B melt-spun magnets through data-driven techniques were explored in [76]. They investigated several machine learning algorithms to build property prediction models of which the model based on Gradient Boosted Regression Trees achieved the best performance for predicting both coercivity and maximum energy product with $R^2 = 0.88$ and $0.89,$ respectively. In [77], a random forest predictor model was used to identify the importance of microstructure characteristics in causing magnetization reversal in ideally structured large-grained Nd$_2$Fe$_{14}$B permanent magnets. The most important features explaining magnetization reversal were the misorientation and the position of the grain within the magnet. [78] investigated the most influential parameters for coercivity in $SmFe_{12}$-based melt-spun ribbons using machine learning approaches. V-addition was found to be the most influential to coercivity. Various supervised machine learning algorithms such as Kernel ridge regression (KRR), support vector regression (SVR), and artificial neural networks (ANN) regression were employed to predict values of coercivity and maximum magnetic energy product of granular NdFeB magnet based on their microstructural attributes such as inter-grain decoupling, average grain size, and misalignment of easy axes from micromagnetic simulations datasets [79]. In [80], a decision tree algorithm was trained to predict nucleation fields calculated by micromagnetic simulation of a quasi-three-dimensional system constructed from a two-dimensional image to identify weak spots in the magnets and observe trends in the nucleation field distribution. In [81], higher-dimensional features such as colour, texture and edge information were used to train a classification model to segment different regions in Kerr microscope images of sintered NdFeB magnet. Subsequently, the segmented images were used to quantify grain sizes and orientation of domain patterns present in each grain.

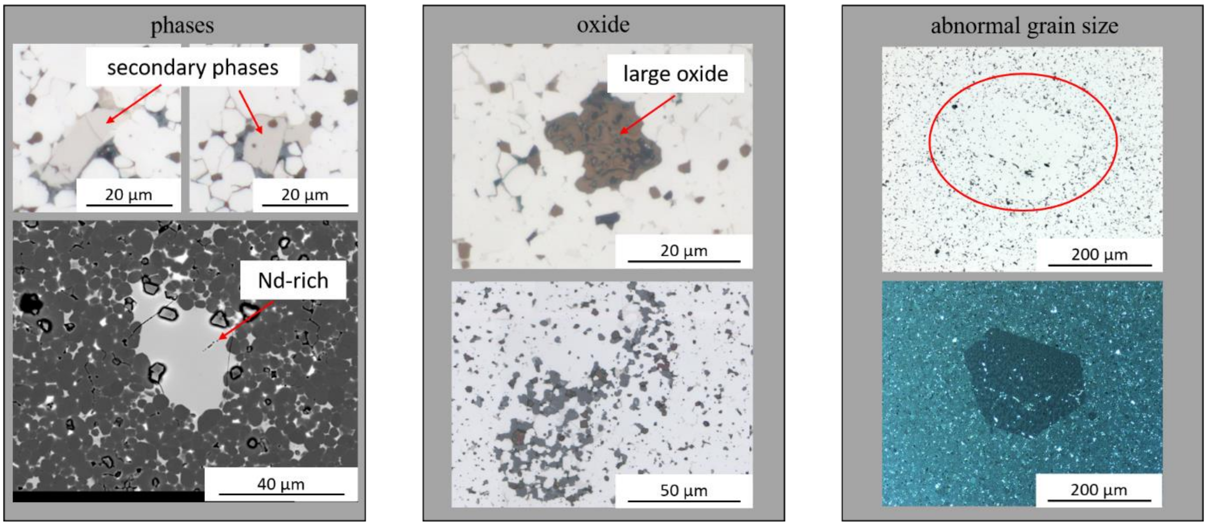


Figure 2-5: Examples of defects found in NdFeB type magnet microstructure.

2.3 Microscopy analysis

There are various microscopy techniques used in a metallographic analysis, such as light optical microscope (LOM), scanning electron microscope (SEM), transmission electron microscope (TEM) and X-ray diffraction techniques. In general, it is usually recommended to first perform light optical microscopy examinations prior to any electron metallographic analysis since those are usually more time-consuming to perform, and the equipment is much more expensive. Conversely, light optical microscopy is fast and can cover a much larger area.

2.3.1 Metallographic preparation

Microscopy analysis has been successfully utilized to examine samples and microstructures for a long time. The aim is to solve a wide range of issues and support product development by evaluating the influence of numerous parameters and for quality assurance in various fields, from medicine to metallography. Similar techniques can be applied to high-resolution images of sectioned Li-ion cells and sintered magnets to facilitate quality assessment through quantitative and qualitative microstructural analysis. Although metallography is a relatively mature field, substantial progress has been made in recent years to automate the preparation of samples and quantify microstructural measurements. The primary objective is the preparation of artefact-free representative samples suitable for microstructural examination. The basic steps for proper metallographic specimen preparation include documentation, sectioning, mounting, planar grinding, polishing, etching and microscopic analysis. However, the particular choice of sample preparation depends on the sample and the focus of the examination, which could be for process optimization, alloy design, reverse engineering, failure analysis or quality assurance. The main focus of this thesis addresses the automation of microscopic analysis for quality assurance using computer vision techniques.

2.3.2 Quantitative microscopy

Metallographic observation has been largely qualitative for most of its history, whereby structures and microstructures are described based on their appearance. However, qualitative microstructure assessment is primarily based on the examiner's knowledge and experience, which can be very subjective. In quantitative metallography, the constituents of the sample’s microstructure are measured to provide more reliable information for an objective assessment. Some of the most basic measurements include estimation of the volume fraction of a phase or constituent, measurement of grain sizes, assessment of the shape of particles, measurements of the size and distribution of features or the relative amount of a structure or phase. Nonetheless, it can be very tedious to manually carry out such quantitative assessments without the risks of fatigue-induced error or bias. Various digital image analysis equipment and software packages have been developed to automate the collection and reporting of quantitative data. However, when it comes to detecting defects, this is typically performed manually, mainly due to the sheer number of possible outcomes. Nevertheless, as a first step towards developing a robust automated defect detection solution, the digitised cross-sections of the specimens investigated in this thesis are visually inspected under the microscope for various inclusions and noticeable irregularities. Then the number of inclusions and anomalies are quantified and sorted into different categories based on visual similarities and expert opinion. This analysis forms the basis of the data used to develop our automated inspection systems in Chapter 3.

2.4 Textural defect detection

The general approach for automated visual inspection in digital image data typically involves texture and colour analysis followed by pattern classification. The former is mainly concerned with extracting and representing features in addition to data perception and modelling, while the latter is concerned with pattern representation, discriminant analysis, or cluster analysis. Texture analysis is one of the most important characteristics for identifying defects or flaws. Texture provides essential and unique information for visual detection and identification systems. It also provides important information for recognition and interpolation. To a large extent, the defect detection problem has been mostly regarded as a texture analysis problem since before any defects can be detected, unique features that describe a particular texture first need to be extracted, which can be a challenging task. Therefore, much research effort has been put into extracting useful texture features over the years.

In most cases, features with large inter-class variations and small intra-class variations are required to differentiate various textures better. After briefly describing the most common approaches used for texture feature extraction, we compare such fixed feature extractors to CNN-based methods. The techniques for extracting texture features for visual inspection tasks can be categorized into four main categories [82]: statistical, structural, spectral and model-based approach. In the following section, a summary of some of the main texture analysis methods applied to defect detection tasks is presented, of which the statistical and spectral-based approaches have been most commonly used for feature extraction.

2.4.1 Standard approaches for texture analysis

Most conventional inspection systems rely extensively on hand-crafted feature extraction. The following describes the most widely used algorithm for general texture feature extraction.

Statistical approaches

Statistical texture analysis methods aim to measure and analyse the spatial distribution of pixel values in a given image. Numerous proposed techniques that fall under this category range from first-order statistics to higher-order statistics, such as histogram statistics, local binary patterns (LBP), co-occurrence matrices, and autocorrelation functions.

Structural approaches

Whereas the statistical approach appears to work well for microtextures, the structural approach works better for macrotextures. Structural approaches (SA) are primarily concerned with the spatial location of the texture elements, also known as texture primitives. In the structural approach, an image is described by the number and types of its primitives and the spatial organization or layout of its primitives. Structural texture is mainly composed of two steps: (1) the extraction of texture elements and (2) the inference of the placement rule [83]. Through specific placement or spatial arrangement rules, the texture is replicated by primitives, resulting in a dynamic texture model. The texture primitives can be individual pixels, line segments, or a uniform grey-scale region.

Spectral approaches

In textures with regular uniform patterns, this regularity can be observed in the frequency domain, such that when there is a defect, the regularity is lost. High-frequency components can be observed in the image when there is a defect, thus making it easier to find in the frequency domain. However, it is typically no longer possible to classify defects once in the spectral domain as the spatial coordinates are lost. Thus, the image needs to be re-transformed into the spatial domain. This approach includes methods such as filter-based approaches, Fourier Transform (FT), Gabor Transform (GT) and Wavelet Transform (WT).

Model-based approaches

The topic of image modelling involves the construction of models or procedures for the image specification. The model can both describe the observed images and synthesise artificial images from the model parameters. Whereas statistical-based methods can be relatively sensitive to noise and spectral-based methods are affected by the absence of local information, model-based methods tend to achieve better performance for various types of defect detection tasks by using a structurally unique model enhanced by parameter learning to project the original texture distribution of image blocks into low-dimensional distribution. They assume some kind of dependence a pixel has on its neighbourhood. Amongst others, some of the most popular model-based methods include autoregressive, random field, and fractal models.

2.4.2 Learning-based approaches for defect detection

The previous section mainly focused on extracting features to discriminate specific texture features or patterns. However, the primary goal of visual inspection is ultimately the detection and classification of these extracted features to perform tasks such as segmentation, quantification, or defects detection. Therefore, an appropriate decision-making scheme needs to be selected, typically referred to as pattern classification. Depending on their processing mechanics, this can be divided into two groups: (1) supervised and (2) unsupervised or semi-supervised classification. For this approach, the system is first trained to recognize specific features which can be obtained from any of the methods discussed in the previous section, such as filtering, thresholding, and other statistical methods, or it can be obtained directly from the image data as is more common nowadays using convolutional networks. A typical computer vision pipeline is as follows: Find points of interest in an image, crop patches around these regions of interest, represent each patch with a sparse local descriptor, and finally combine the descriptors into a representation of the image. While this can be a powerful technique, it has a few drawbacks. First, a lot of expert knowledge and effort is required to extract meaningful and representative features from the images. Second, it is often susceptible to slight deviations such as changes in lighting conditions, background colours and noise. Furthermore, since the features identified via handcrafted techniques are not sufficiently discriminative, they are often challenging to scale, especially for complex datasets. However, learning the features directly from the data helps make the system more applicable to domains where prior knowledge of acceptable and defect patterns is not well consolidated.

Visual inspection through supervised classification

In supervised classification, the detectable features are predefined, and the classifier must be trained to recognize those features. Supervised learning aims to model a conditional distribution between the input vector (surface image or extracted features) and target vector (defect label). Different types of classifiers exist for recognising patterns, such as linear discriminators, distance-based classifiers, Bayesian classifiers, neural networks, and others. K-Nearest Neighbour [84], support vector machine (SVM) [85], [86], and neural network are classical examples commonly used for defect detection [87]–[89]. We will discuss neural networks at length in Section 2.5.

Visual inspection through unsupervised classification

A large amount of labelled training data is often required to train a supervised machine learning model, of which the collection of both positive (examples with defects) and negative samples (examples without defects) can be complicated and time-consuming. On the other hand, unsupervised approaches that do not require any labelling of the training data often suffer from high false alarms and low detection rates [90]. Nevertheless, normal samples are often much easier to obtain for many real-world applications than anomalous samples, which are very expensive to collect. The automated detection of anomalies has become a vital issue in many industrial applications due to the ever-increasing amount of data that needs to be analysed. These problems are characterized by instances of interest being heavily underrepresented in the data, often accounting for less than one case in a thousand [91]. Some classic examples of anomaly detection problems include intrusion detection in computer networks [92], detection of fraud in credit card transactions [93], detection of oil spills in satellite images [94] and detection of rare diseases in health care systems [95]. Thus, anomaly detection refers to the problem of discovering patterns in a data set that do not conform to the expected behaviour, which can otherwise be referred to as outliers or novelties. Furthermore, since anomalies are rare and diverse, obtaining a labelled dataset that represents all possible outcomes is not feasible. Therefore, a successful approach for anomaly detection would be to learn a model of the normal class with the assumption that the training data consists entirely of normal observations, such that if an observation deviates from the learned model, it is classified as an anomaly. Addressing this task using machine learning methods requires a redesign of the learning strategy given the reduced amount of flawed cases available for training compared to the normal instances, which have been shown in several cases to hinder the performance of traditional classification algorithms.

There are three main approaches for building one-class classifiers: density estimation methods, boundary methods, and reconstruction methods [96]. However, most of the successful approaches for anomaly detection are designed for low-dimensional datasets and suffer significant challenges as the dimensions of the data increase. High-dimensional datasets pose significant challenges for anomaly detection due to several factors [97]: (1) the number of potential feature subspaces grows exponentially as the input dimension increases, resulting in an exponential search space. (2) Every point in a high-dimensional space appears as an anomaly. In other words, given enough alternative subspaces, it is possible to find at least one feature subspace for each point where it appears as an anomaly. (3) The true anomalies are effectively masked by the high proportion of irrelevant features creating noise in the input data. As a result, direct applications of these approaches to high-dimensional datasets may produce undesired results. The extremely high dimensionality of image data makes the underlying properties difficult to capture by density estimation or boundary methods.

One widely used technique to address this challenge is by mapping high-dimensional data into lower-dimensional subspace using various dimensionality reduction techniques such as principal component analysis [98], linear discriminant analysis [99] and machine learning methods [97], in order to efficiently process the resulting data with standard detection algorithms. For high-dimensional data such as images, autoencoder neural networks have shown promising results in anomaly detection problems [100]. This type of network consists of an encoder network, which can perform linear and nonlinear transformations from the input into a lower-dimensional latent representation, which is then decoded back into the original image by the decoder network. As a result, autoencoders do not require label information since the input image also represents the target output. Through the reconstruction of the original image, the network is able to learn representative features that can generalize to the reconstruction of images similar to those in the training dataset with low reconstruction errors. On the contrary, images that deviate significantly from those observed during training will produce higher residual errors. Therefore, this reconstruction error can be used as an anomaly score to distinguish normal images from outliers.

Autoencoder learns to map an input image $x \in \mathcal{X} = \mathbb{R}^n$ to an output image $x' \in \mathcal{X},$ by reconstructing the original image from an intrinsic lower dimensional latent representation. It comprises two modules, an encoder network $f : \mathcal{X} \rightarrow Z$ and a decoder network $g : Z \rightarrow \mathcal{X},$ both implemented as a multi-layer neural network. Both modules are jointly trained to compute $x' = g\big(f(x)\big),$ where the output of the encoder $z = f(x) \in Z = \mathbb{R}^m$ $(m \ll n)$ is a low-dimensional latent representation of $x.$ The low-dimensional latent representation is a bottleneck that prevents the autoencoder from learning a trivial identity function that directly maps the input to the output. The autoencoder is trained to minimize the reconstruction error $L(x, x'),$ which is usually the mean squared error between the pixel values of the original image and the reconstructed image. Once the training is complete, anomalies can be detected by comparing $L(x, x')$ to a decision threshold $T_{rec},$ where all images y with $L\Big(y, g\big(f(y)\big)\Big) > T_{rec}$ are classified as anomalies. The value of T_{rec} is typically chosen based on the distribution of all reconstruction errors L_{train} on the training set X_{train} which can be either the maximum reconstruction error $T_{rec} = max_{x \in X_{train}} L(x, x')$ or a large percentile $T_{rec} = p_{0.95}(L(x, x') | x \in X_{train})$ for the 95th percentile, which is generally more robust. By this, we assume that all training examples should only consist of normal observations and should have low reconstruction errors. Therefore, for the autoencoder to be used for detecting anomalies, it needs to learn an implicit model of the data distribution seen during training, such that it can reconstruct previously unseen normal images with minimal error while reconstructing anomalous images with maximum error. However, only the first criteria can be directly optimised if the training dataset consists of just the normal cases and the autoencoder is trained to minimise the reconstruction error.

Secondly, autoencoder-based anomaly detection methods are susceptible to the slightest presence of anomalies in the training set. That is, given enough training iterations, the autoencoder will learn to reconstruct anomalous observations as well as normal ones if even a small number of anomalies leak into the training data. Techniques like reducing the model capacity and early stopping have been proposed to prevent the network from being able to reconstruct the anomalies so that it only focuses on reconstructing the majority class. However, these techniques also prevent the autoencoder from learning a model that can accurately reconstruct the majority of normal training observations and may lead to higher false detection rates. On the other hand, generative models aim to learn the true data distribution in order to generate new data points that are similar to the training set, albeit with some variations. Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are the two most common and efficient generative techniques for processing image data. Thus, by leveraging the ability to learn the input data distribution, these techniques can be more effective at identifying anomalies on high-dimensional and complex datasets.

2.5 Neural networks and deep learning

In what follows, we review the basic building blocks for building a deep learning network, starting from the simplest linear model, the perceptron, which consists of a single neuron from which we build the multilayer perceptron. For a task such as image classification or object localization, we require more specialized models. A large part of the following section is dedicated to convolutional neural networks that we use to classify defect images and later for object detection and image generation.

2.5.1 Artificial neural networks

The simplest neural network is the perceptron, consisting of only a single neuron. Conceptually, the perceptron functions similarly to a biological neuron. A biological neuron receives electrical signals from its dendrites, modulates the signals and if the total strength of the input signal exceeds a certain threshold, it fires an output signal through its synapses which are fed as input to another neuron. Similarly, the artificial neuron performs two consecutive functions to model the phenomenon in the biological neuron. First, the weighted sum of the inputs is calculated to represent the total strength of the input signals, and then an activation function is applied to the result to determine the output value of the neuron. However, even though the inspiration from biology is evident, it would be misleading to overemphasize the connection between biological neurons and artificial neurons. In the perceptron diagram shown in Figure 2-6, the neuron k receives m input parameters x_j. The neuron also has m weight parameters w_{kj} which is used to represents the importance of different input data points. The weight parameters often include a bias term with a fixed value of 1. A linear combination of the inputs and weights is fed to an activation function φ that produces the output y_k of the neuron:

$$y_k = \varphi(\sum_{j=0}^{m} w_{kj} x_j). \tag{2.2}$$

The neuron is trained by repeatedly updating the weights to produce the desired output for each input controlled by the types of activation functions.

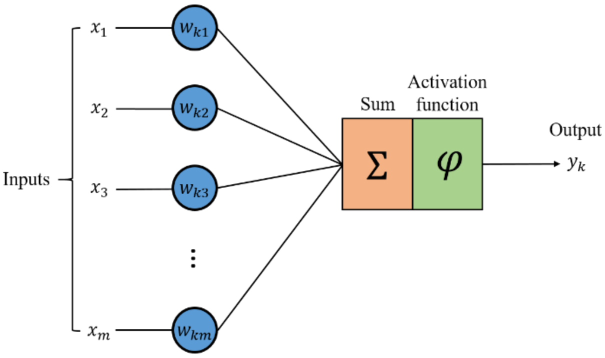


Figure 2-6: An artificial neuron

A neural network consists of many such neurons structured into layers to predict an output. For example, a simple feed-forward network illustrated in Figure 2-7 consists of a series of fully connected layers where each output layer of neurons is fed as input to neurons in the subsequent layer. This type of architecture is also often referred to as a multi-layer perceptron (MLP). Thus, a multi-layer network is typically composed of an input, an output and one or more intermediate layers, also known as hidden layers. Each layer consists of a linear function followed by a non-linear transformation. The output layer converts the activations in the hidden layer to an output value such as a probability distribution for classification tasks or continuous values for regression. There are other types of network connections, for example, in convolutional networks, which make use of parameter sharing and thus have much fewer parameters than fully connected networks.

In other words, a neural network can be described as a mathematical model for information processing, which is defined by:

1. The neural network architecture: This describes the set of connections between the neurons (feed-forward, convolutional or recurrent), the number of neurons in each layer, and the number of layers.
2. The learning protocol: Typically referred to as model training, where the weights between the neurons are updated. Over the years, various methods for updating the weights have been developed, from energy level training to back-propagation.

3. The activity function: The capacity of the neural network to approximate any function, especially non-convex, is essentially due to the non-linear activation function. It determines the final output of each neuron and under what conditions it will activate.

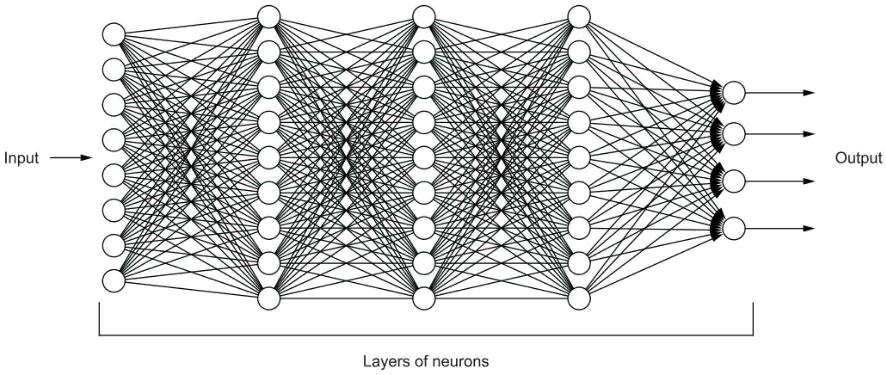


Figure 2-7: A fully connected multi-layer neural network.

The following function describes a multi-class logistic regression:

$$f(x) = Wx + b$$
$$g(y) = softmax(y) \tag{2.3}$$

where $W \in \mathbb{R}^{C \times d}, x \in \mathbb{R}^{d}, b \in \mathbb{R}^{C}, y \in \mathbb{R}^{C}, C$ is the number of classes and d is the dimensionality of the input. From now, W is used to denote a matrix of weights, while $\theta \ni W, b$ is the set of parameters of the model. Logistic regression can be seen as a composition of the function $g\big(f(x)\big)$ where $f(\cdot)$ is the affine function and $g(\cdot)$ is the activation function.

A neural network is a construct of multiple such affine functions interleaved with non-linear activation functions. The ReLU [101] is a common activation used in-between multiple layers, while the sigmoid and softmax are typically used at the output layer of the network to obtain a Bernoulli and categorical distribution, respectively. A network with only one hidden layer and a softmax output can thus be represented as follow:

$$h = \sigma_1(W_1 x + b_1)$$
$$y = softmax(W_2 h + b_2) \tag{2.4}$$

where σ_1 is the activation function of the first hidden layer. Each layer in the network is parameterized with its own weight matrix W and bias vector b. However, in some cases, layers can set their parameters to be the same values, known as weight tying or sharing. This is typically used to introduce an inductive bias into the model, leading to better generalization.

2.5.2 Loss functions

To evaluate how well the network is performing, we need to compare the output from the model with the ground truth label. The loss function provides a method for measuring the error in the predicted output value in order to minimize it. We are searching for the set of weights that achieve the lowest score. Please note that we may interchangeably use the terms loss function, cost function, object function, and error function but still refer to the same thing. The choice of the loss function is determined by the type of problem we are trying to solve, and more importantly, it is directly related to the activation function used in the output layer of the neural network. In this section, we present some of the most commonly used loss functions to train deep neural networks for regression and classification:

L1 loss or mean absolute error loss can be defined as:

$$\mathcal{L}(y, y') = \sum_{i} |y_i - y_i'| \tag{2.5}$$

L2 loss or mean squared error loss is a common loss function used for regression. It is defined as follows:

$$\mathcal{L}(y, y') = \sum_{i} (y_i - y_i')^2 \tag{2.6}$$

Cross-entropy loss. For classification problems, the mean squared error (MSE) used to be the primary choice of loss function. However, a model trained using this objective suffered from saturation and slow learning when using sigmoid or softmax activations. The cross-entropy loss has mostly replaced the MSE loss function for most multi-class classification problems. The cross-entropy between the empirical conditional probability $p(y|x)$ and the probability of the model $\hat{p}(y_i|x;\theta)$ for each example x is:

$$H(p, \hat{p}; x) = -\sum_{i=1}^{C} p(y_i|x) \log \hat{p}(y_i|x;\theta) \tag{2.7}$$

for binary classification, this can be simplified to:

$$H(p, \hat{p}; x) = -(1-y)\log(1-\hat{y}) - y\log\hat{y} \tag{2.8}$$

as the loss function $J(\theta)$, the average cross-entropy is minimized over all examples in the data:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} H(p, \hat{p}; x_i) \tag{2.9}$$

Maximum likelihood estimation (MLE). While the squared error is generally easy to compute, it is not the most appropriate in some cases. A more general principle for estimation is the maximum likelihood estimation. It provides a framework for finding the best statistical estimates of parameters from the training data. An MLE model is defined as a function

$p_{model}(x;\theta)$ that maps an input x to a probability using a set of parameters θ. Consider a set of m examples $X = \{x^{(1)}, \dots, x^{(m)}\}$ drawn independently from the true data distribution $p(x)$ which is unknown, we approximate the true probability $p(x)$ with the probability $\hat{p}_{data}(x)$.

The maximum likelihood estimator is defined as:

$$\theta_{MLE} = \arg\max_{\theta} p_{model}(Y|X;\theta)$$

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} p_{model}(x_i;\theta) \tag{2.10}$$

Many of the probabilities in the product can be small and lead to numerical instabilities. Thus to obtain a more convenient optimization problem, the product can be converted to a sum by taking the logarithm of the likelihood function, which does not change the arg max [2].

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \sum_{i=1}^{n} \log p_{model}(x_i;\theta). \tag{2.11}$$

The arg max does not change if we divide the log-likelihood function by n or (any constant value). Then we can obtain an expectation with respect to $\hat{p}_{data}$ as defined by the training data.

$$\theta_{MLE} = \arg\max_{\theta} \mathbf{E}_{x\sim\hat{p}_{data}} p_{model}(y_i \mid x_i;\theta). \tag{2.12}$$

The maximum likelihood can therefore be seen as an attempt to make the model distribution match the empirical distribution $\hat{p}_{data}$.

Kullback-Leibler divergence. The maximum likelihood can also be seen as minimizing the dissimilarity between the empirical distribution $\hat{p}_{data}$ and the model distribution, p_{model} [2]. This dissimilarity is defined by the Kullback-Leibler (KL) divergence [102] as:

$$D_{KL}(\hat{P}_{data} \parallel P_{model}) = \mathbf{E}_{x\sim\hat{p}_{data}}[\log \hat{p}_{data}(x) - \log p_{model}(y \mid x;\theta)] \tag{2.13}$$

where **E** is the expectation. Since the term on the left is only a function of the data generating distribution and not a function of the model, the model can be trained to minimize the KL divergence by only minimizing the term on the right-hand side:

$$-\mathbf{E}_{x\sim\hat{p}_{data}}[\log p_{model}(y \mid x;\theta)]. \tag{2.14}$$

The minimization of the KL divergence is equivalent to the maximization of the MLE since minimizing a negative term is the same as maximizing the term; hence this objective is the same as the MLE objective in Equation (2.12).

$$\hat{\theta}_{MLE} = \arg\min_{\theta} -\mathbf{E}_{x\sim\hat{p}_{data}}[\log p_{model}(x_i;\theta)]. \tag{2.15}$$

Furthermore, minimizing the KL divergence is equivalent to minimizing the cross-entropy (Equation (2.9)) between the empirical distribution $\hat{p}_{data}$ and the model distribution p_{model}. Hence, cross entropy is prevalent in machine learning and the objective function that is most commonly used in neural networks. As a result, we will frequently use it throughout this thesis.

Huber loss. Lastly, the Huber loss is a piecewise loss function that is less sensitive to outliers than squared error:

$$\mathcal{L}_{\delta} = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{for } |y-\hat{y}| < \delta \\ \delta\left(|y-\hat{y}| - \frac{1}{2}\delta\right), & \text{otherwise} \end{cases} \tag{2.16}$$

It is quadratic for small differences between the label y and the prediction $\hat{y}$ and linear for large values. It is used later for bounding box regression.

2.5.3 Forward and backpropagation

A neural network learns a mapping of input to output from the training data. Due to the large number of parameters needed in any practically useful model, it is impossible to calculate the exact weights for solving a particular problem analytically. Instead, the problem is cast as an optimization problem where an algorithm is used to search the space of possible sets of weights that sufficiently solves the problem. The neural network is typically trained using an optimization algorithm such as stochastic gradient descent, and the weights are updated using the backpropagation of error algorithm [103].

When training a neural network, the first step involves a forward pass of information from the input, propagating through all the hidden units at each layer to the output. This is called forward propagation. All the intermediate variables from the input to the output are calculated and stored within the model. During training, in the forward pass, the weighted sum of each unit’s input is calculated, the activation function is applied, and the output is passed to the next layer until the final output layer is reached and the network predicts an outcome. The error rate is the difference between the predicted and actual values. For feed-forward networks and other deep neural networks, the most efficient way to train the model is to compute a cost function based on the misclassification error and minimize the loss function with respect to the model's parameters. Therefore, since the error is a function of the weights in the network, we want to minimize the error with respect to the weights. For an input value $\boldsymbol{x} \in \mathbb{R}^d$ and assuming there is no bias term for simplicity, mathematically, this can be written as:

$$z = W_1 x \tag{2.17}$$

where $W_1 \in \mathbb{R}^{h\times d}, z \in \mathbb{R}^h$ is the weight parameter of the hidden layer and the intermediate variable, respectively. If we apply an activate σ we obtain a hidden layer variable $h \in \mathbb{R}^h$

$$h = \sigma(z) \tag{2.18}$$

if we also assume the parameter of the output layer $W_2 \in \mathbb{R}^{q \times h}$, we obtain the output variable with a vector of q,

$$o = W_2 h \tag{2.19}$$

lastly, assuming a loss function l and the target label y, the loss term for a single data point can be calculated as,

$$L = l(o, y) \tag{2.20}$$

the $\ell_2$norm regularization term of the network weights can be written as,

$$s = \frac{\lambda}{2}(||W_1||_F^2 + \ ||W_2||_F^2), \tag{2.21}$$

finally, the network's regularized loss is given as,

$$J = L + s \tag{2.22}$$

where J is referred to as the objective function.

Backpropagation and partial derivatives

Many training algorithms involve an iterative procedure for minimising an error function, with adjustments to the weights being made in a sequence of steps. As each model consists of multiple layers, calculating the gradient of the loss function with respect to each parameter is non-trivial. A dynamic programming algorithm known as backpropagation is used to compute the gradient. In the first stage, the derivatives of the error function with respect to the weights are evaluated, and in the second stage, the derivatives are used to compute the weight adjustments. The backpropagation algorithm provides a computationally efficient method for evaluating these derivatives. Furthermore, the second stage of the weight adjustment using the calculated derivative can be achieved using a variety of optimization schemes besides the simple gradient descent, which we will discuss in the next section. The goal is to minimize the error rate calculated after the forward pass. Mathematically, the error is formulated in terms of a loss function $E(\hat{y}, y)$ where $\hat{y}$ is the predicted label, and y is the ground truth. The loss function is minimized based on the weights and biases in the network.

If we consider a simple linear model where the outputs y_k are linear combinations of the input variables x_i:

$$y_k = \sum_i w_{ki} x_i \tag{2.23}$$

In the forward pass, each unit computes a weighted sum of its inputs in the form:

$$a_j = \sum_i w_{ji} z_i \tag{2.24}$$

Where z_i is the activation of a unit connecting unit j , and w_{ji} is the associated weight. A nonlinear activation function $h(\cdot)$ transforms the sum in Equation (2.24) above to give the activation z_j of unit j as:

$$z_j = h(a_j) \tag{2.25}$$

Through the successive application of Equation (2.24) and Equation (2.25), each input vector in the training set can be propagated through the network. Now, assuming the error function E_n for a particular input pattern n, and note that E_n depends on the weight w_{ji} only through the summed input a_j. We can therefore apply the chain rule for partial derivatives to give:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{2.26}$$

which can be rewritten as:

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{2.27}$$

where the error signal δ_j for the hidden unit j is

$$\delta_j = \frac{\partial E_n}{\partial a_j} \text{ and } \frac{\partial a_j}{\partial w_{ji}} = z_i \tag{2.28}$$

and for the output units, we have

$$\delta_k = y_k - t_k \tag{2.29}$$

to evaluate the $\delta's$ for the hidden units, we can use the chain rule for partial derivatives.

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{2.30}$$

Finally, the backpropagation formula can be obtained as

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{2.31}$$

2.5.4 Optimization methods

Unlike linear regression with the mean square error loss, there is typically no closed-form solution to obtain the optimal weights for most losses. Instead, the model's error is minimized iteratively using an algorithm known as gradient descent. Gradient descent is one of the most popular and efficient methods for training neural networks. Given an objective function

$J(\theta; X, Y)$, where $\theta \in \mathbb{R}^d$ is a vector of network parameters, X the input vector and Y the corresponding target values. Gradient descent updates the parameters in the opposite direction of the gradient $\nabla_\theta J(\theta)$ of the loss function. The gradient is the vector containing all the partial derivatives $\frac{\delta}{\delta_{\theta i}} J(\theta)$. Where the i-th element of the gradient is the partial derivative of $J(\theta)$ with respect to θ_i. Depending on the size of the data used to calculate the gradient, there are three primary variants of the gradient descent algorithm: 1) batch gradient descent, 2) stochastic gradient descent, and 3) mini-batch gradient descent.

Batch gradient descent. In batch gradient descent, the entire training set is used to update the parameters of the model according to

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.32}$$

where η is the learning rate that determines the magnitude of each parameter update.

In practice, η is one of the most important hyperparameters that need to be correctly tuned to guarantee convergence of the algorithm. If the value is set too low, the training will take a long time to converge, and if it is set too high, the gradient might become very large. However, there are a few techniques for finding an appropriate learning rate, such as those proposed in [104] and [105].

Stochastic gradient descent (SGD). Alternatively, stochastic gradient descent is when each sample from the dataset is used to make individual updates to the model parameters. The update rule is given as:

$$\theta = \theta - \eta \nabla_\theta J(\theta; x_i; y_i) \tag{2.33}$$

where x_i and y_i are the i^{th} training samples. While this is cheaper to compute, the resulting gradient estimate is a lot noisier due to the stochasticity.

Mini-batch gradient descent. A more common approach is to take a mini-batch of n examples from the training set to compute the gradients and update the model parameters. This is known as mini-batch gradient descent or stochastic gradient descent with mini-batches, and it usually results in a much smoother loss curve than SGD.

$$\theta = \theta - \eta \nabla_\theta J(\theta; x_{(i:i+n)}; y_{(i:i+n)}) \tag{2.34}$$

The size of the mini-batch n can range from 2 to a few hundred depending on the available physical memory of the hardware, which enables the training of large models on vast datasets with millions of examples.

Adaptive methods

Even though mini-batch gradient descent works well in practice and is one of the primary methods for training neural networks, it has a few weaknesses. 1) It does not allow automatic learning rate adjustment during training, which means the same learning rate is used for all its parameters. 2) It does not remember its previous steps, which makes it susceptible to getting trapped in saddle points due to near-zero gradients around them. In order to overcome the limitations of mini-batch gradient descent, several other optimization methods and improvements have been proposed, such as Nesterov Accelerated Gradient [106], Root Mean Square Propagation and Adaptive Moment Estimation [107].

2.5.5 Dealing with class imbalance

Over the last two decades, several class imbalance learning techniques have been developed. They can be divided into two main categories: data-level and algorithm-level [108]. Resampling approaches [109]–[112] are the most common data-level techniques that aim to balance the distribution of classes in the training data. Furthermore, depending on the method used for balancing class distribution, the resampling technique can be further divided into three subgroups [113].

- Over-sampling method: this aims to increase the minority class samples. Randomly duplicating minority samples and generating synthetic minority samples are two commonly used methods.
- Under-sampling method: some of the majority class samples are discarded from the dataset in this method. The simplest form of this method, random under-sampling, involves randomly removing samples from the majority class.
- Hybrid method: is a combination of both over-sampling and under-sampling methods.

The cost-sensitive learning method [114]–[116] is a typical algorithm-level approach for learning from an imbalanced dataset. In this approach, a higher misclassification cost is assigned to the minority class, essentially forcing the classifier to focus more attention on examples from the minority class. Several approaches [117]–[120] have been developed that implement new loss functions for handling imbalanced datasets, which have shown some good results. Nevertheless, some drawbacks are associated with both cost-sensitive learning and resampling approaches. For instance, in the resampling approaches, randomly over-sampling the minority class can easily result in over-fitting, while important information may be discarded in the under-sampling of the majority class. On the other hand, a cost-sensitive learning approach is more computationally efficient. However, assigning an appropriate cost for each class can be difficult. We evaluate the effectiveness of the resampling and the cost-sensitive approaches for our defect detection task and compare the performances to when no modification is made to compensate for the class imbalance. For the resampling approach, we compare both the over-sampling and under-sampling techniques, and for the cost-sensitive approach, we selected the weighted cross-entropy loss function and focal loss function.

The focal loss function [120] was originally designed for object detection tasks to tackle the extreme foreground-background class imbalance encountered during the training of dense detectors. This is accomplished by reshaping the standard cross-entropy loss to down-weigh the loss associated with well-classified examples to allow the detector to focus training on the sparse set of harder examples. This prevents a large amount of easy negative examples from overwhelming the network during training. The focal loss function has been widely used in various applications, from mitosis detection in medical images [121] to semantic segmentation of high-resolution aerial images [122].

2.6 Convolutional neural networks

Convolutional neural networks (CNNs or convnets) are neural networks with one or more convolutional layers. They are hierarchical models that alternate between two basic operations, convolution and subsampling, inspired by the local receptive fields and the hierarchical structure of simple and complex cells in the primary visual cortex [123]. Convnets are primarily used for solving fundamental image processing problems in computer vision, such as image classification, object detection, localisation and segmentation. However, it can be more generally used to handle data with some spatial topology (e.g. videos, sound spectrograms in speech processing, character sequence in texts, or 3D voxel data). In many of these cases, the input dimensionality is often high. For example, a 256×256 colour image contains 196,608 pixels ($256 \times 256 \times 3$) for 3 colour channels. If each pixel intensity of this image is input separately to a fully connected network, each neuron requires 196,608 weights. For a 512×512 colour image, each neuron will require 786,432 weight connections. As a result, the network's overall number of free parameters greatly increases as the image size increases. This is not only inefficient in terms of the number of parameters and processing time, but such large models will also result in overfitting.

Convolutional neural networks originally formulated in [124], [125] and later in [3] were developed precisely to address this issue. They are aware of the spatial layout of the input and use specific local connectivity and weight sharing to significantly reduce the number of free parameters. This means each node only connects to spatially adjacent nodes in the next layer, making them useful for image processing, as it is reasonable to assume that each pixel exhibits some correlation with neighbouring pixels. Furthermore, because convnets share weights, the number of free parameters does not grow proportionally to the input dimension. Therefore, they are well suited for normal-sized images and are responsible for many recent computer vision breakthroughs. The parameters of a convolutional layer called receptive fields or convolutional kernels are replicated across the entire visual field, making it particularly suited for local feature extraction. This fundamental characteristic gives convnets two important properties: 1) They learn *translation-invariant* patterns, which means that after learning a specific pattern in one part of an image, a convnet can recognize it anywhere else in the image. This makes convnets data efficient when processing images. 2) They learn *spatial hierarchies* of patterns. This means that each successive layer learns patterns made of the features of the preceding layer, allowing convnets to learn increasingly complex and abstract visual concepts efficiently. In contrast, a fully connected network fails to consider this kind of structure. A schematic illustration of a convnet is shown in Figure 2-8. In this thesis, we investigate the use of convolutional networks for accurately detecting defects in the microstructure of Li-ion batteries and sintered NdFeB magnets.

2.6.1 Layers

We will now describe some layers and models commonly applied to computer vision tasks that will be used throughout this thesis.

Convolutional layers. The core computational building block of a convnet is the convolutional layer which takes an input tensor and produces an output tensor by convolving the input with a set of kernels. The convolutional layer implements a similar concept to the convolution operation in mathematics, more commonly seen in signal processing and Fourier analysis. A 2D convolution of two functions, I and K is defined as:

$$S(i,j) = \sum_{m}\sum_{n} X(m,n)K(i-m,j-n) \tag{2.35}$$

Where X is the input, and K is the convolution kernel. The negative signs for the m and n indices of K indicate that the kernel is flipped both in the vertical and horizontal directions and therefore translated over the input. However, flipped kernels do not hold any significance in the learning process of convolutional networks; therefore, in practice, this is most often implemented using a non-flipped convolution operation known as *cross-correlation* [2]. Which can be expressed as:

$$S(i,j) = \sum_{m}\sum_{n} X(m,n)K(i+m,j+n) \tag{2.36}$$

where the kernel K operates only on a small region of the input, defined by its size $(m \times n)$.

Suppose that our input is a colour image represented by a 3D tensor $X.$ The convolution operates by sliding the kernel across all spatial positions of the input tensor and computing the dot product between a small patch of X and the kernel K at each position. The convolution operation for images can therefore be represented as:

$$S(i,j) = (X * K)(i,j) = \sum_{l=1}^{K_1}\sum_{m=1}^{K_2}\sum_{n=1}^{C} X(l,m,n)K(i+l\,,j+m,n) \tag{2.37}$$

The input image is a 3D tensor of size $[D_1, D_2, C]$ where D_1 is the height, D_2 is the weight, and C is the number of channels. The corresponding kernel is also a 3D tensor of size $[K_1, K_2, P]$, where K_1, K_2 and P are the height, weight and output channels, respectively. The result of the convolution operation is called an *activation or feature map*. For example, consider a colour image of width and height of 256 – i.e. $256 \times 256 \times 3$ tensor X and a $5 \times 5 \times 3$ kernel $K,$ which contains 75 parameters that will be automatically learnt. The result of the convolution operation will be an activation map of dimensions $252 \times 252 \times 3$ (252 is the number of unique positions that a filter of 5 elements can be placed over an input of size 256). It is also common to pad the input with a border of zeros to control the output dimension. In this case, if we pad the borders with zeros of thickness 2, this will result in a $256 \times 256 \times 3$ activation map. Similarly, a kernel of size $5 \times 5 \times n$ will produce an activation map of size $252 \times 252 \times$

n with zero padding of the borders. It is also possible to perform convolution with some strides. The stride is the number of pixels the kernel shifts at each step. A larger stride will result in smaller sized output. For example, applying the $5 \times 5 \times 3$ kernel to the $256 \times 256 \times 3$ input tensor with a stride of 2 and no padding will instead result in an output activation map of size $126 \times 126 \times 3$. Systematically applying the same kernel across the entire image allows the kernel to discover similar features anywhere else in the image; this capability is commonly referred to as translation invariance. For instance, if a kernel is designed to detect a specific type of texture, then the application of that kernel across the entire image enables that particular texture to be detected anywhere in the image. The parameters that make up the filters are learnt through backpropagation.

In general, a convolutional layer for images:

- Accepts a tensor of size $W_1 \times H_1 \times D_1$ (i.e. assuming input tensor with three spatial dimensions)
- Requires four hyperparameters: The number of out channels K, the kernel size F, the stride S, and the amount of zero padding on the borders of the input P.
- Produces an output with a spatial dimension $W_2 \times H_2 \times D_2$, where

$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1,\ H_2 = \frac{(H_1 - F + 2P)}{S} + 1, \text{ and } D_2 = K \qquad (2.38)$$

- The number of parameters in each kernel is $F \times F \times D_1$ and the total number of parameters in the convolutional layers is thus $(F \times F \times D_1) \times K$ weights $+$ K biases.

Pooling layers. In a convolutional network, a pooling layer is typically used to provide some degree of translation invariance to slightly different inputs and to reduce the dimension of the feature response from preceding convolutional layers. It also helps to minimize the likelihood of overfitting. The pooling layer operates on each feature map separately and downsamples them spatially by summarizing the presence of some features in patches of the activation map. There are two commonly used types of pooling operation which determine how the downsampled feature map is aggregated: Average pooling calculates the average value within a patch on the activation map. Maximum pooling (max pooling) calculates the maximum value for each patch of the activation map. A pooling layer is defined by specifying two parameters, the size and the stride. The size parameter specifies the window over which the pooling operation is performed, and the stride specifies the number of pixels by which the pooling window shifts after each operation, similar to that of the convolution layers. It is common to observe a 2×2 window using a stride of 2 and applying the max pooling operation. Max pooling is typically preferred as it avoids cancelling negative elements and prevents blurring of the activation maps.

Fully Connected layers (FC). The final layers of a convnet are typically made up of one or more fully connected layers. This is the standard feed-forward neural network where all the input neurons are connected to all the output neurons. The main difference between a fully connected layer and a convolutional layer is that the fully connected layer learns global patterns in its feature space, whereas a convolutional layer learns local patterns. The output feature maps from the convolutional layers are flatted into a 1D vector before feeding the fully connected layers. The objective of the fully connected layer is to take activation maps from the convolutional layers and predict the class labels. These layers have a non-linear activation (hidden layers) or a softmax activation (output layer) to output a probability of class predictions. They are used to perform a non-linear combination of features and make final predictions by the network. Thus, while the convolutional layers are used for feature extraction, the fully connected layers act as classification layers. The parameters of the fully connected layers are specified according to the number of inputs and outputs required. The number of units in the output layers is equal to the number of classes. Thus, for an input vector of size m and output vector of size $n,$ a $m \times x$ parameter matrix is required. As a result, due to these highly dense connections, a few fully connected layers can easily account for a large subset of the network's weights.

Residual connections. One of the major benefits of neural networks is the ability to represent various complex functions simply by stacking more layers. However, after a certain depth, very deep networks often have a gradient signal that goes to zero during training, commonly known as *vanishing gradient.* If we consider a convolutional layer as mapping $H(x)$ of input x from one feature space to another. Instead of directly finding the mapping $H(x)$ we find another mapping $F(x) = H(x) - x,$ which can be rewritten as $H(x) = F(x) + x.$ The idea is to add a *residual or skip connection* [126] which allows information to flow more easily by preserving the characteristics of the original vector x before its transformation by some layers $F(x)$ and simply performing the sum $F(x) + x.$ The authors claim that, since the gradient is additive, it is unlikely to vanish for a large number of layers.

ConvNet architecture. The basic building block of a convolutional neural network consists primarily of the components discussed in the previous sections. Typically, convnets are built using blocks of convolutional layers (Conv), activation function (AF), and possibly pooling layers (Pool) to control the computational complexity of the architecture, followed by a series of fully connected layers (FC). It is also possible to use strided convolutions instead of pooling layers to reduce the spatial resolution of the feature maps. For image classification, the convolution, ReLU and pooling layers are typically interleaved to increase the expressive power of the network. Having two or three sets of Conv-ReLU combinations is common before having a max pooling layer. This entire pattern can be repeated a few times to create deep convolutional networks. As the depth of the network increases, recent architecture designs also use skip connections between layers [126]. A convnet architecture with two convolutional layers, two pooling layers and two fully connected layers might take the form illustrated in Figure 2-8. In theory, the convolutional layers closer to the input learn to recognize low-level features of the image. For instance, the first layer may learn to detect edges and corners, while the second may learn to detect more complex shapes formed by combining different edges, such as rectangles and circles, etc. The layers closer to the output would learn to combine the features generated in the previous layer to recognize much more complicated objects [127]. Thus, the power of convolutional neural networks lies in the ability to construct complex and interpretable visual features by piecing together these primitive representations layer by layer.

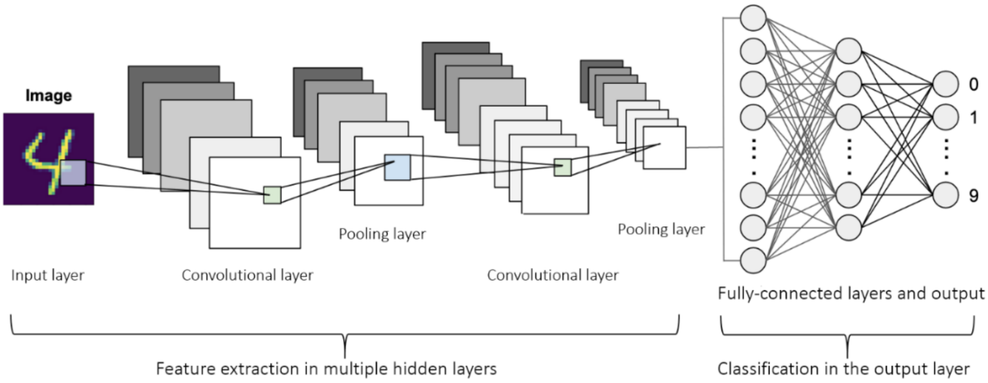


Image
Input layer
Convolutional layer
Pooling layer
Convolutional layer
Pooling layer
Fully-connected layers and output
0
1
9
Feature extraction in multiple hidden layers
Classification in the output layer

Figure 2-8: An example of a convolutional neural network.

2.6.2 Fully convolutional networks

So far, the neural network architectures we have seen are usually designed for image classification and feature learning tasks. In order to adapt these models for other computer vision problems such as semantic segmentation, instance segmentation, depth estimation or optical flow, one must configure the architecture in such a way that it outputs images instead of class labels or feature vectors. In other words, the model needs to learn a pixel-to-pixel transformation from an input image to an output image. For example, we need to determine the class of each pixel within an image in semantic segmentation tasks. This type of architecture is often referred to as a *fully convolutional network* [128] because it is built only from locally connected components, such as convolution, pooling and upsampling layers. Since there are no dense layers in this type of architecture, the number of parameters and computation time is reduced. Furthermore, the network can operate on images of various sizes. In image classification, an input image is downsampled via convolution with strides, or pooling layers, then passes through a fully connected layer to output the class probabilities. However, to obtain a segmentation map, the spatial resolution of the original image needs to be recovered to predict the class label for each pixel. Therefore, this usually involves two parts: 1) a downsampling path to capture semantic or contextual information and 2) an upsampling path to recover precise spatial information. There is usually a lot of low-level information shared between the input and output for many image translation problems lost in the pooling or downsampling layers. Thus, a skip connection is often used to shuffle this information directly across the network. This is done by concatenating or element-wise addition of feature maps from the downsampling path with feature maps from the upsampling path. Upsampling of the feature maps can be achieved via bilinear interpolation or convolution with fractional input strides known as deconvolution [129].

2.6.3 Variational autoencoder

Variational autoencoder (VAE) emerged as an important method for unsupervised representation learning of complex distributions, which can also be used as a generative model.

It was simultaneously proposed in [130] and [131]. In a classical autoencoder [132], the input vector is mapped to a latent vector space through the encoder module, which is then decoded back to the original dimension by the decoder. In other words, given an input x the autoencoder tries to generate an output $\hat{x}$ that is as similar as possible to x. We are usually not interested in the decoded output for most applications but rather in the internal representations learned by the network. However, this thesis will fully utilise the decoded output to detect defects using an unsupervised learning approach. Initially, the VAE was introduced as a constrained version of the autoencoder that learns an approximation of the underlying data distribution. Thus, instead of compressing the input data into a fixed latent code, it turns the data into the parameters of a statistical distribution that can be sampled to generate new data. Like a standard autoencoder, a variational autoencoder consists of an encoder and a decoder trained to minimize the reconstruction error between the input data and the decoded output. However, instead of encoding the input as a single point in latent space, it is encoded as a regularized latent distribution, constrained to be close to a standard Gaussian distribution. The network is trained as follows:

- the input data is first encoded as a distribution over the latent space
- then we sample a point from this distribution
- and decode the sampled point back to the original input
- then the reconstruction error is calculated and backpropagated through the network.

The stochasticity of this process forces the latent space to encode meaningful representation across it, such that any point sampled from the latent space can be decoded to a valid output. In practice, the encoder is trained to learn the parameters of a latent unit Gaussian distribution and return the mean and covariance matrix that describes this Gaussian distribution that can best regenerate the input data. The final loss function is the sum of the reconstruction error and the regularization loss, which is the Kullback-Leiber (KL) divergence between the reconstructed latent variable and the Gaussian distribution. The KL divergence is used as a regularization technique to enforce two main properties: *continuity* (i.e. two close points in the latent space should decode to similar outputs) and *completeness* (i.e. any point sampled from the latent space of the chosen distribution should produce meaningful content when decoded).

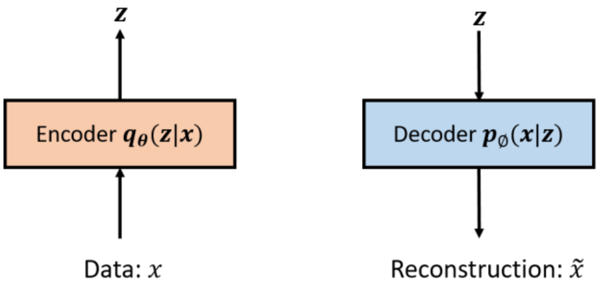


Figure 2-9: The encoder compresses the input data into a latent space z, and the decoder reconstructs the data from the hidden representation.

We now provide a mathematical definition as follows: consider input data x generated from a latent variable z which is not directly observable. Let us denote the probabilistic encoder $q_\theta(z \mid x)$ which has parameters θ, takes input x and outputs hidden representation z. We denote the probabilistic decoder $p_\phi(x \mid z)$ with parameters ϕ, which takes the hidden representation z and outputs the parameter to the probability distribution of the data (see Figure 2-9). The loss function l_i for data point x_i is:

$$l_i(\theta, \phi) = -E_{z\sim q\theta}(z|x_i)\left[\log p_\phi(x_i \mid z)\right] + KL(q_\theta(z \mid x_i) \mid\mid p(z)). \tag{2.39}$$

The first term is the expected negative log-likelihood or reconstruction loss of the i-th data point, used to encourage the decoder to reconstruct the original input data. The second term is the KL divergence between the encoder's distribution $q_\theta(z \mid x)$ and a standard Gaussian distribution $p(z) = \mathcal{N}(0, 1)$. However, a reparameterization trick is required to ensure that the network is differentiable in practice. This consists of setting $z = \mu + \sigma \odot \epsilon$, where $z \sim \mathcal{N}(\mu, \sigma)$ and $\epsilon \sim \mathcal{N}(0, 1)$. It allows the randomness of a normally distributed random variable z to be pushed into ϵ, which is sampled from a standard normal distribution.

2.6.4 Generative adversarial networks

Generative adversarial networks (GANs) are another form of generative modelling based on deep neural networks. The GAN architecture was proposed in [133] to leverage the power of discriminative learning to train good generative models. The main idea behind GANs is that if we have a good enough data generating model, we should not be able to distinguish fake data (generated data) from real data. This is similar to the two-sample test in statistics which tries to determine whether the difference between two datasets $X = \{x_1, \ldots x_n\}$ and $X' = \{x_1', \ldots, x_n'\}$ is statistically significant. The GAN model architecture involves two sub-models trained simultaneously Figure 2-10: 1) a generator model used to generate new examples from the problem domain. 2) A discriminator model is used to classify examples as either real (from the original data distribution) or fake (a generated example). That is, the generator model is given an input z drawn randomly from a Gaussian distribution and outputs a sample x from an implicit probability distribution P_g. While the discriminator model is a classifier which takes an input x and outputs a binary class label which determines if x is from the original data distribution P_{data} or from P_g. The real examples are obtained from the training set, and the generated examples are produced by the generator model. The training process, therefore, involves the generator attempting to "fool" the discriminator with the generated examples, and the discriminator network is constantly adapting to newly generated fake data, which in turn is used to improve the generator model in a two-player *minimax* game with the value function $V(G, D)$ defined as:

$$\min_G \max_D V(G, D) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_g(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right] \tag{2.40}$$

where $D(x)$ is the discriminator's estimate of the probability for real images, E_x is the expected value over all real data instances, $G(z)$ is the generator's output when given input z, $D(G(z))$ is the discriminator's estimate of the probability for generated images and E_z is the expected value over all generated fake instances $G(z)$. This can be summarized as follows: the discriminator seeks to maximize the average of the log probability of real images and the log of the inverse probability for generated images. While the generator seeks to minimize the log of the inverse probability predicted by the discriminator for generated images. Which basically encourages the generator to generate realistic samples with low probabilities of being classified as fake. However, during the early stages of training, when the generator's performance is poor, the discriminator can very easily distinguish generated samples from real samples with high confidence causing the loss function for the generator to saturate and the generator to stop updating. The original GAN publication [133] suggests one approach to solving the saturation problem is to modify the generator loss so that the generator tries to maximize $\log D(G(z))$ instead of minimizing $\log(1 - D(G(z))$. This essentially means that instead of the generator seeking to minimize the probability of generated images being predicted as fake, it now seeks to maximize the probability of those images being predicted as real. This new objective function results in better gradient information for updating the generator's weights and more stable training dynamics.

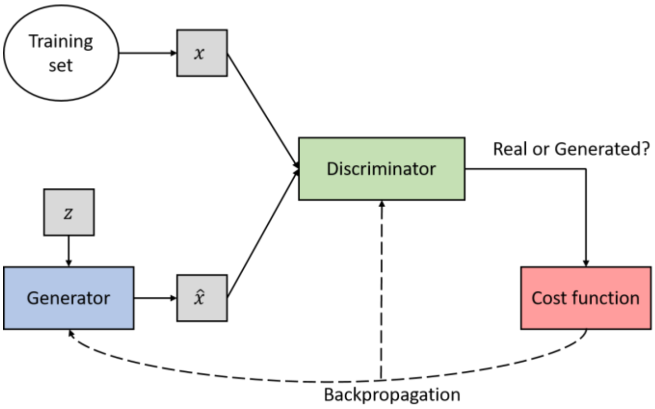


Figure 2-10: Generative adversarial network framework.

The most successful application of GANs has been on image data, where a convolutional neural network is used as the generator and the discriminator model. In this case, the generator learns a compressed representation of the set of images from the training data distribution and can generate plausible images. Since the originally proposed framework, many other types of GAN architectures have been invented for tackling domain-specific problems, such as Deep Convolutional GAN (DCGAN) [134], Conditional GAN (CGAN) [135], CycleGAN [136], InfoGAN [137] and many more. GANs have also been used for many real practical applications. For example, Multi-Condition GAN (MC-GAN) [138] was used for conditional synthesis to perform a text-to-image transformation. In cases with limited training data, GANs can be used to generate many additional examples for data augmentation. Other use cases include style transfer and image manipulation, e.g. inpainting, face ageing [139] and generating high-resolution versions of the input images [140].

Conditional GAN. In the original implementation of the GAN, there was no control over the modes of data being generated. However, in many cases, such as the text-to-image or image-to-image synthesis, we are interested in generating outputs of a specific type for a given input signal. This can be achieved by conditioning the model on additional information such as class labels or data from other modalities to direct the data generating process [135]. We can therefore extend the generative model of a GAN to a conditional model if the generator and the discriminator are both conditioned on some auxiliary information y. The input to the generator model thus becomes the random vector z and the additional input y. The discriminator is also provided with an input that is either real or generated and the additional input y. Suppose the conditional input was a class label; the discriminator would expect that the input would be from that class, which encourages the generator to generate examples from the same class. The conditioning can be performed by feeding y into both the generator and the discriminator through an additional input layer, as illustrated in Figure 2-11. The value function from Equation (2.40) of the two-player minimax game therefore becomes:

$$\min_G \max_D V(G,D) = E_{x\sim p_{data}(x)}[\log D(x \mid y)] + E_{z\sim p_g(z)}\left[\log\left(1 - D\left(G(z \mid y)\right)\right)\right] \quad (2.41)$$

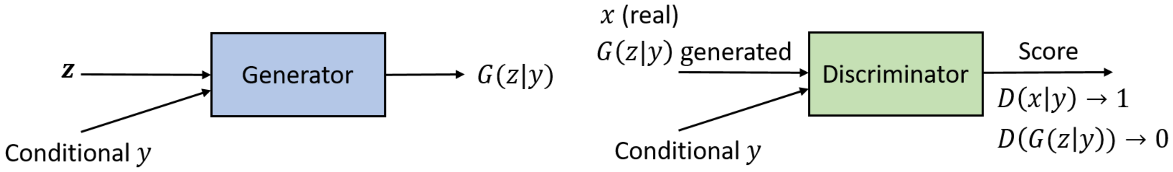


Figure 2-11: Conditional generative adversarial network.

2.7 Convolutional object detection and localisation

Due to the rapid developments in deep learning, particularly convolutional neural networks for image classification, more powerful techniques have emerged in recent years to solve object detection tasks that address the problems facing traditional approaches. These methods can learn semantic, high-level, deeper features to realize significant performance improvements over previous methods. Object detection aims to determine where objects are located in an image (object localisation) and which category the objects belong to (object classification). The general approach for CNN-based object detection models can be categorized into two main types: (1) a two-stage approach, which involves generating region proposals and classifying each proposal into different object classes. This includes methods such as R-CNN, SPP-net, Fast R-CNN, Faster R-CNN, R-FCN, FPN and Mask R-CNN etc. (2) One-stage approach that regards the object detection task as a regression or classification problem. This includes MultiBox, AttentionNet, YOLO, SSD, RetinaNet, DSSD etc. This section compares different CNN-based object detection methods relevant to this thesis.

Two-stage Detectors

OverFeat [141] is one of the most prominent approaches among the earlier attempts at object detection using CNN. This method inspired most modern region-based object detectors since it pioneered the integration of image classification and regression tasks into one CNN model. To achieve this, a CNN is used to perform image classification at different locations on multiple scales of an image through a sliding window, and then it performs bounding box regression on the topmost feature map after obtaining the confidence of each object category. First, a CNN model is trained for image classification then the top classifier network is replaced with a class-specific regression network, which is used to refine the boundaries of bounding boxes at each spatial location. Next, a greedy merging algorithm aggregates the resulting class scores and regressed bounding boxes. Despite the success of this method, it had a few drawbacks. For instance, there was significant redundancy in the number of processed windows containing any objects, which considerably increased the running time. Nevertheless, the combination of image classification and regression from OverFeat has been extended in modern object detection frameworks, where the main idea is to localize objects in an image by combining region proposals with CNNs using multiple regions of interest.

2.7.1 Region-based Convolutional Neural Networks

In [142], the first region-based object detector called Regions with CNN features (R-CNN) was proposed. The R-CNN architecture consists of three modules. In the first step, regions of interest (RoI or region proposals) are generated using an external algorithm to scan the input image for possible regions containing objects. The region proposals are category independent bounding boxes with a high probability of containing an object of interest. There are various methods for generating region proposals such as selective search [143], Edge Boxes [144], objectness [145], Constrained Parametric Min-Cuts (CPMC) [146], Category-Independent Object Proposals [147], Multi-Scale Combinatorial Grouping [148] etc. Selective search is most commonly used among these methods because it is fast and has a high recall.

R-CNN adopts selective search to generate around 2000 category independent region proposals from each input image. The selective search algorithm is based on computing hierarchical grouping of similar regions based on colour, texture, size and shape compatibility to provide more accurate candidate boxes of arbitrary sizes to reduce the search space for object detection. In the second module, each sub-image contained in the region proposal is warped or cropped into a fixed dimension to match the input size of the pre-trained CNN used to extract a fixed length (4096 dimensional) feature vector. Lastly, the third module scores the extracted feature vectors using a set of pre-trained class-specific linear support vector machines (SVMs) into a set of positive regions or negative (background) regions. The scored regions are then filtered using a greedy non-maximum Suppression (NMS) algorithm and adjusted with bounding box regression to produce the final bounding boxes of objects in the input image.

Although the R-CNN showed significant improvements over traditional methods, it also suffered some drawbacks that have been improved upon by later methods. Training the R-CNN involves multiple stages. First, the convolutional network is fine-tuned on object proposals then the classification layer is replaced with SVMs to fit the extracted features before bounding box regressors are trained. This makes the pipeline difficult to train. The process of generating region proposals is time-consuming (around 2 seconds to generate 2000 region proposals), and a lot of the proposals are redundant. Another major drawback is the fixed input size of region proposals, which is achieved by warping or cropping the sub-image irrespective of its original size or aspect ratio to a pre-defined dimension before input to the CNN. Lastly, the approach is very computationally expensive in terms of storage requirements and processing time. This results from storing features extracted from each region proposal (~2000 per image) for later processing during training and passing warped sub-images individually through a deep network. Consequently, this usually requires many days of computation and hundreds of gigabytes of storage space. At test time, features are individually extracted from each object proposal in the test image, resulting in a slow inference speed taking about 47 seconds per image.

2.7.2 Fast Region-based Convolutional Neural Network

An improved and more practical method for object detection called Fast Region-Based Convolutional Neural Network (Fast R-CNN) [149] was proposed to address some of the limitations of R-CNN. Instead of extracting features for each region proposal individually as in R-CNN, the entire image is processed with the CNN to produce feature maps. The method receives an image and a set of object proposals computed using an external method as input. Then for each region proposal, the corresponding location on the feature map is extracted and resized to a fixed-length feature vector using a region of interest (RoI) pooling layer. Each feature vector forms the input to a sequence of fully connected layers which branches into two output layers: a softmax layer responsible for producing probabilities for all $K+1$ categories (K objects classes plus one 'background') and a real-valued layer that encodes refined bounding box co-ordinates computed using regression. Consequently, the Fast R-CNN architecture jointly trains the CNN, classifier and bounding box regressor end-to-end in a single model using backpropagation. This results in more efficient use of storage space, computation time, and improvements in detection accuracy. The parameters of the model, excluding the method for generating region proposals, are optimized jointly for classification and bounding-box regression using a multi-task loss L defined as:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v) \tag{2.42}$$

where $L_{cls}(p, u) = -\log p_u$ calculates the log loss for ground truth class label, $u \in 0,1, \dots, K$; by convention, the catch-all background class has $u = 0$. The discrete probability distribution (per RoI) $p = (p_0, \dots, p_K)$ computed by a softmax over the $K+1$ outputs from the final fully connected layer. $L_{loc}(t^u, v)$ is defined over the predicted offsets $t^u = \left(t_x^u, t_y^u, t_w^u, t_h^u\right)$ and ground-truth bounding-box regression targets $v = \left(v_x, v_y, v_w, v_h\right)$, where x, y, w, h denote the two coordinates of the box centre, width, and height, respectively. $[u \geq 1]$ evaluates to 1 when $u \geq 1$ and otherwise 0 is employed to exclude all "background" RoIs. The hyperparameter λ controls the balance between the two task losses. For more robustness against outliers and to eliminate sensitivity to exploding gradient, a smooth L_1 loss is adopted for bounding-box regression, which is defined as follows:

$$L_{loc}(t^u, v) = \sum_{i \epsilon x,y,w,h} smooth_{L1}(t_i^u - v_i) \tag{2.43}$$

where

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{2.44}$$

During the training of the Fast R-CNN model, mini-batches of N images are sampled randomly and R/N RoIs are sampled from each mini-batch, where R is the number of RoIs. If the intersection over union (IoU) of an RoI with a ground-truth box is over 0.5, the RoI samples are assigned to a class; otherwise, they are assigned to the background class. Furthermore, RoIs from the same image share computation and memory usage in the forward and backward pass. As a result, Fast R-CNN can achieve a much shorter detection time than R-CNN, taking less than a second on a recent GPU, mainly due to using the same feature map for each RoI. However, with a decrease in detection time, the overall time spent on computation now depends mainly on the performance of the method used for generating region proposals. Consequently, RoI generation can easily become a computational bottleneck that affects the network's overall performance.

2.7.3 Faster Region-based Convolutional Neural Network

R-CNN and Fast R-CNN both rely on an external method such as selective search or Edge Boxes to generate a candidate pool of region proposals. However, as mentioned previously, the region proposal computation is time-consuming and effectively becomes the computational bottleneck in improving performance. Both object detection methods use CPU based selective search algorithm, which takes around 2 seconds per image, consuming as much running time as the detection network. Faster Region-based Convolutional Neural Network (Faster R-CNN) [150] was developed as a successor to Fast R-CNN to address the computational bottleneck caused by using an external region proposal algorithm. Faster R-CNN is an integrated method that uses shared convolutional layers to generate regional proposals and object detection. The external region proposal method in Fast R-CNN is replaced with a *Region Proposal Network* (RPN), which proposes and refines region proposal as part of the training process, and the Fast R-CNN architecture is used as the detection network. Since RPN shares convolutional layers with the object detection network, generating regional proposals becomes almost cost-free. The authors mainly observed that feature maps used by object detection networks could also generate region proposals. This reduces the region proposal time from 2s to 10ms per image and sharing the convolutional layers between region proposal and detection results in an overall improvement in feature representation. Therefore, we can consider the Faster R-CNN as the combination of a region proposal network (RPN) and a Fast R-CNN object detector, unified in a single model which can be trained end-to-end or by alternating between fine-tuning for region proposal task and then fine-tuning for object detection task while keeping the proposal fixed.

This results in a unified network that shares convolutional features for both tasks. The architecture of Faster R-CNN is shown in Figure 2-13.

Region Proposal Network (RPN). The RPN is a fully convolutional network that predicts bounding box coordinates and objectness scores at each position of an image, which is subsequently used as region proposals for the object detector network. It is specifically trained end-to-end for generating high-quality region proposals. Whilst other object detection networks such as OverFeat, Spatial Pyramid Pooling, Fast R-CNN etc., make use of a pyramid of scaled images or a pyramid of different filter sizes, RPN uses a pyramid of regression reference scheme. This enables Faster R-CNN to deal with detection windows of different shapes and sizes and achieve a much better running speed performance when trained on single-scale images since it avoids enumerating images or filters of multiple scales or aspect ratios.

First, an input image is fed into the backbone convolutional neural network to extract feature maps. The RPN takes these feature maps as input and outputs a set of rectangular object proposals, each associated with an objectness score used to determine whether the proposal contains an object or not. Computation is shared with the Fast R-CNN detector by employing the same convolutional layers of the backbone network for feature extraction. For every point in the output feature map, the RPN needs to predict whether an object is present in the input image at this corresponding location and estimate its size. This is done by sliding a small CNN over the output feature map from the last shared convolutional layer, producing multiple object proposals at each sliding-window location (see Figure 2-12). The maximum number of possible region proposals at each location is denoted as k and parametrized relative to k so called *anchors* (references boxes). These anchors are designed to accelerate and improve region proposals at each location by pre-defining possible objects of various scales and aspect ratios. An anchor is centred at each sliding window location, with each anchor having 3 scales $(128^2, 256^2, 512^2)$ and 3 aspect ratios (1:1, 1:2, 2:1), yielding a total of $k = 9$ anchors at each sliding position. As the RPN moves through each pixel in the output feature map, we check if the anchor contains any objects and refine the coordinates of the anchors to produce bounding boxes as object proposals. Next, a 3×3 convolution with 512 output units is applied to the output feature map, thus mapping each sliding window to a lower-dimensional feature space. This produces a 512-d feature map for every anchor box, which is fed into two sibling layers: a 1×1 convolution layer with $4k$ output units for bounding-box regression (reg) and a 1×1 convolution layer with $2k$ output units for object classification (cls). For a convolutional feature map of size $H \times W,$ there are $(H \times W) \times k$ anchors in total. The regression layer has an output of size $(H \times W \times 36)$ which give the 4 regression coefficient (centre coordinates, height and width) of each of the 9 anchors for every point in the feature map. These coefficients are used to refine the coordinates of the anchors containing any objects. Likewise, the classification layer has an output size $(H \times W \times 18)$ which gives the probability of a proposal containing an object or not at each point in the feature map within all 9 of the anchors at that point.

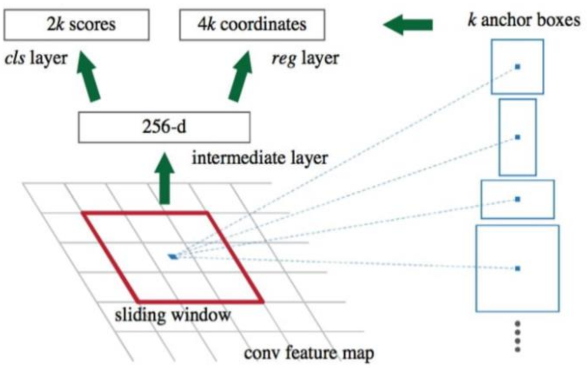


Figure 2-12: How the RPN works [150].

Loss functions. Each anchor box is assigned a binary label (object or not object). An anchor is assigned a positive label if it satisfies either of two conditions: 1) the anchor has the highest IoU (intersection over union) overlap with a ground-truth box. 2) The anchor has an IoU higher than 0.7 with any ground-truth box. It is also possible that the same ground-truth box may cause multiple anchors to be assigned positive labels. The first condition is adopted in rare cases where the second condition fails to find any positive samples. On the other hand, when an anchor has an IoU less than 0.3 for all ground-truth boxes, it is assigned a negative label. The remaining anchors (neither positive nor negative) are ignored for RPN training. During training, all anchors that cross the image boundary are disregarded, avoiding any contribution to the total loss. If the boundary-crossing anchors are not ignored, they introduce large error terms in the objective, which are difficult to correct, thus preventing the training from converging. Each mini-batch (256 samples) for training the RPN is obtained from a single image. If all the anchors from the image are sampled, this will bias negative samples in the learning process. Therefore 128 positive and 128 negative samples are randomly chosen to form a mini-batch, and in case of an insufficient number of positive samples, the mini-batch is padded with additional negative samples. With this, the training loss for the RPN for an image is given by:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*) \tag{2.45}$$

where i is the index of the anchor in the mini-batch. p_i is the predicted probability of anchor i being an object from the classification branch and $p^*$is the ground-truth label (1 for positive anchor and 0 for negative one). L_{cls} is the log loss over two classes (object vs not object) for classification. The regression loss L_{reg} is activated only for positive anchors i.e. $p_i^* = 1$ and disabled otherwise. t_i is a vector representing the output prediction of the regression layer, which consists of four variables $[t_x, t_y, t_w, t_h]$ and t_i^* represents the coordinates of the ground-truth box associated with a positive anchor, calculated as:

$$t_x^* = (x^* - x_a)/w_a,\;\; t_y^* = (y^* - y_a)/h_a,\;\; t_w^* = \log(w^*/w_a),\;\; t_h^* = \log(h^*/h_a) \tag{2.46}$$

where $x, y, w,$ and h represent the anchor box's centre coordinates, width and height, respectively. x^*, x_a represent the coordinates of the ground-truth box and the anchor box (likewise for y, w, h). The classification loss is normalized by the size of the mini-batch N_{cls} and the regression loss is normalized by the number of anchor locations N_{reg}. The two terms are weighted by a balancing parameter λ. The learned regressor output t_i is applied at test time to its corresponding anchor box, and the parameters x, y, w and h for the predicted object proposal can be calculated from:

$$t_x = (x - x_a)/w_a, \;\; t_y = (y - y_a)/h_a, \;\; t_w = \log(w/w_a), \;\; t_h = \log(h/h_a) \qquad (2.47)$$

Non-maximum suppression (NMS) is adopted for some highly overlapping region proposals to reduce redundancy based on their class score. The IoU threshold is set to 0.7 for NMS. The bounding box proposals from the RPN are used to pool features from the output convolution feature map, which forms the input for the Fast R-CNN detector network.

Training methods

Separately training the RPN and Fast R-CNN networks will cause them to learn different features in their convolutional layers. Therefore, a training method is required to facilitate sharing convolutional layers between the two networks, rather than learning two isolated networks. This section looks at different techniques for training Faster R-CNN as a unified network.

Alternate training. In this approach, the RPN is trained end-to-end to generate region proposals. Then the proposals are used to train the detector network. The tuned object detector network is again used to initialize the RPN, and the process is repeated alternatingly.

Approximate joint training. This approach merges the RPN and the Fast R-CNN network into a unified network, as shown in Figure 2-13. When training the Fast R-CNN detector, during the forward pass, the RPN generates region proposals which are treated as fixed, pre-computed proposals, and during the backward pass, the backward propagated signal from the RPN loss and the Fast R-CNN loss are combined for the shared layers. However, by doing so, we ignore the derivatives with respect to the coordinates of the generated region proposals, which are also the network's response, thus making it an approximate solution. Nonetheless, the approach is easier to implement and reduces training time by 25-50% compared with the alternate training method while producing comparable results. Therefore, we adopt this method for all related experiments in this thesis.

Non-approximate joint training. Given that the region proposals generated by the RPN are also functions of the input, but the gradients with respect to the bounding boxes' coordinates are ignored in the approximate joint training approach, a differentiable RoI pooling layer with respect to the bounding boxes' coordinates is required in a non-approximate training approach while the remaining training procedure is the same as above.

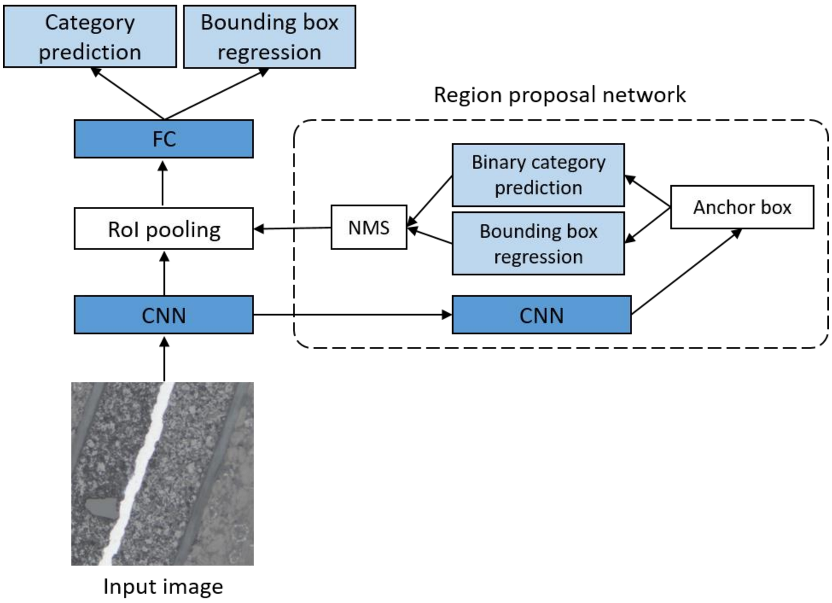


Figure 2-13: Illustration of the Faster R-CNN model architecture.

2.7.4 Single Shot MultiBox Detector (SSD)

One-stage Detectors

All the above methods for object detection we previously discussed (R-CNN, Fast R-CNN and Faster R-CNN) are all region-based frameworks, whereby one part of the network is dedicated to generating region proposals while the other part classifies these proposals. The detection typically happens in two stages: First, the model proposes a candidate pool of regions of interest (RoIs) through a selective search or region proposal network. Features are extracted from the input image with a deep CNN, and then classification and bounding box regression are performed on the region candidates. These steps are usually performed separately, and even in the case of Faster R-CNN, an alternate training scheme is often required for the RPN and detector networks to share convolutional parameters. Even though they can achieve high detection accuracy, the computational cost of handling different components may become a major drawback for some real-time applications due to the low frame rate. Combining these two stages into one network is a different approach for performing object detection known as a one-stage approach. The region proposal is completely eliminated in the one-stage approach, and detection is run directly over a dense sampling of possible locations. Thus, classification and regression are done in a single shot using dense sampling of pre-defined boxes to locate objects at various scales and aspect ratios. Since the whole detection pipeline is a single network, optimization can be performed directly end-to-end based on detection performance. Consequently, this approach is much faster (both for training and inference) and easier to implement. However, this is often at the cost of slightly worse detection accuracy.

The Single Shot MultiBox Detector (SSD) [151] is one of the most commonly used one-stage detectors. The main components consist of a base network and several multiscale convolutional feature layers added in series to the end of the base network, as illustrated in Figure 2-14. The base network, typically a deep CNN, is used to extract features from the input image. The original implementation uses a truncated VGG-16 architecture [152], replacing the fully connected layers with a set of auxiliary convolutional layers to progressively decrease the size of the input feature map at each subsequent layer while extracting features at multiple scales. However, in recent implementations, the base network is commonly replaced with variations of the ResNet architecture [126]. The network fuses predictions from multiple feature maps with different resolutions to process objects of various sizes. Similarly to Faster R-CNN, SSD utilizes a set of default boxes (anchors) with different aspect ratios and scales to discretize the output space of bounding boxes.

Each anchor box has a fixed size and position in relation to its corresponding cell, and the whole feature map is tiled with pre-defined anchor boxes. SSD generates a different number of anchor boxes with different aspect ratios based on the output feature size of the base network and each multiscale feature block to detect objects of various sizes. The multiscale feature layers are responsible for predicting the offsets to the default boxes (anchors) and their associated confidence scores for every location of the feature map. They can be seen as a pyramid representation of the input image at different scales where each multiscale feature block reduces the height and width of the feature map provided by the previous layers (i.e. the size of the feature map is halved). The base network can be designed such that the output feature map has a larger height and width. This enables more anchor boxes to be generated based on this feature map, thus allowing smaller objects to be detected. Each element in the feature map is used to expand the receptive field on the input image so that multiscale feature blocks closer to the output have a smaller coarse-grained feature map and generate fewer anchor boxes. As a result, they have a larger receptive field and are better suited for detecting larger objects. On the other hand, earlier layers have larger fine-grained feature maps, which are good at capturing small objects. Since feature maps at different levels have receptive fields of different sizes, we rescale the anchor boxes on different levels so that one feature map is only responsible for detecting objects at a particular scale. At each location (i, j) of the l-th feature layer $l = 1, \dots, L$ of size $m \times n, i = 1, \dots n, j = 1, \dots m,$ the anchor boxes are defined according to a unique linear scale proportional to the layer level as:

$$s_l = s_{min} + \frac{s_{max} - s_{min}}{L - 1}(l - 1) \tag{2.48}$$

and 5 aspect ratios $r \in \{1, 2, 3, 1/2, 1/3\}$ with an additional scale $s_l' = \sqrt{s_l s_{l+1}}$ when the aspect ratio is 1, this gives a total of 6 anchor boxes per feature cell. The width w_l^r and height h_l^r are given by $s_l\sqrt{r}$ and $s_l/\sqrt{r}$ respectively with centre location $\left(x_l^i, y_l^j\right) = \left(\frac{i+0.5}{m}, \frac{j+0.5}{n}\right)$.

For each of k anchor boxes, the model outputs 4 offsets and c class probabilities by applying 3×3 convolutional filters. Therefore, for a given feature map of size $m \times n$, we need $kmn(c + 4)$ prediction filters. During training, SSD selects one box from the set of anchor boxes by matching each ground-truth box with the default box with the highest overlap according to a specified threshold. The network is trained end-to-end to minimize the cost function, which is a weighted sum of the localization loss $\mathcal{L}_{loc}$ and confidence loss $\mathcal{L}_{conf}$ according to:

$$\mathcal{L}(x,c,l,g) = \frac{1}{N}\left(\mathcal{L}_{conf}(x,c) + \alpha\mathcal{L}_{loc}(x,l,g)\right) \tag{2.49}$$

where N is the number of matched default boxes. If $N = 0$ (no default box was found), the total loss is set to 0. α balances the weights between the two losses (picked by cross-validation). The localization loss is a smooth L1 loss between the parameters of the predicted box l and the ground truth box g. SSD also regresses to offsets for the centre (cx, cy) of the default bounding box d, with width w and height h, similar to Faster R-CNN bounding box regression.

$$\begin{gathered}\mathcal{L}_{loc}(x,l,g) = \sum_{i\in Pos}^{N} \sum_{m\in\{cx,cy,w,h\}} x_{ij}^{k}\, smooth_{L1}(l_i^m - \hat{g}_j^m) \\ \hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \\ \hat{g}_j^{w} = \log(g_j^w/d_i^w) \qquad \hat{g}_j^{h} = \log(g_j^h/d_i^h)\end{gathered} \tag{2.50}$$

The confidence loss is the softmax loss over multiple class confidences c:

$$\mathcal{L}_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p}\log\left(\hat{c}_i^p\right) - \sum_{i\in Neg}\log(\hat{c}_i^0) \tag{2.51}$$

where

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \tag{2.52}$$

where x_{ij}^p indicates whether the i-th default box and the j-th ground truth box are matched for an object in class p. Pos is the set of matched (positive) bounding boxes, and Neg is the set of negative examples selected through hard negative mining. During training, as most of the default boxes will have low IoUs, negative examples outweigh the number of positive ones. Therefore, all the negative default boxes are sorted according to their confidence loss, and the model picks the top candidates with the highest confidence for training so that the ratio between negative and positive samples is at most 3:1. Conceptually, SSD is simpler than the region-based detectors as it eliminates proposal generation and any subsequent feature resampling, making it faster at test time.

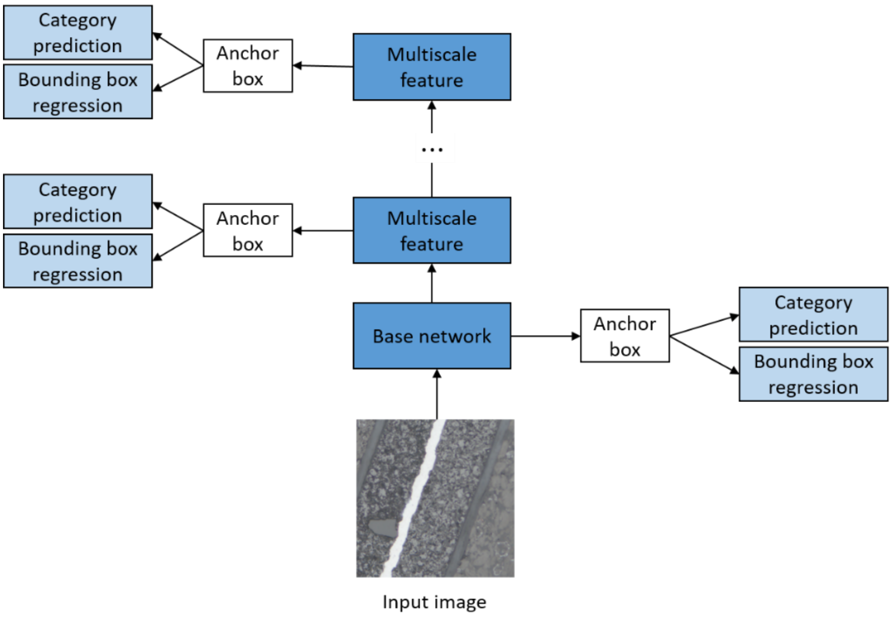


Figure 2-14: Illustration of the SSD model architecture.

Chapter 3

Experimental procedure and proposed solutions

3.1 Investigated lithium-ion battery specimens

All investigations were performed on commercial lithium-ion round cells of type 18650, typically used in power tools and plug-in electric vehicles. For developing the supervised defect detection models, all training samples were obtained from specimens with the cathode comprising lithium, nickel, manganese, and cobalt oxide (NMC) coating, while the anode coating consists mainly of graphite. The specific mixing ratio of the cathode materials is $LiNi_{0.6}Mn_{0.2}Co_{0.2}O_{2}$, often abbreviated as NMC-622. Table 3-1 describes all the battery specimens used in our experiments.

Table 3-1: Listing of all the examined battery cells and their properties.

Sample designation	Manufacturer	Model	Nominal capacity mAh	Positive electrode material	Negative electrode material
INSP_018	LG Chem	INR18650MG1	2850	NMC	Carbon
INSP_026	LG Chem	ICR18650B4	2600	NMC	Carbon
INSP_030	LG Chem	INR18650MG1	2850	NMC	Carbon
INSP_033	LG Chem	ICR18650B4	2600	NMC	Carbon
INSP_038	LG Chem	ICR18650B4	2600	NMC	Carbon
INSP_042	LG Chem	INR18650HB6	1500	NMC	Carbon
INSP_044	LG Chem	ICR18650B4	2600	NMC	Carbon
INSP_047	LG Chem	ICR18650B4	2600	NMC	Carbon
INSP_051	LG Chem	ICR18650B4	2600	NMC	Carbon
INSP_007	Samsung	INR18650-30Q	3000	NCA	Carbon
INSP_004	Sanyo	UR18650W2	1500	NMC, MnO	Carbon
INSP_001	Sony	US18650VTC5A	2600	NCA	Carbon, SiO

3.2 Lithium-ion battery sample preparation

Before commencing with the sample preparation, the open-circuit voltage of the cell is measured and documented, then discharged with an incandescent lamp at $\leq 2.5\ V$. For the light microscope examination, cross-sections were prepared based on the detailed preparation instructions described in [153]. First, the cell is fixed in a bespoke sample holder and sectioned across the marked indications, as illustrated in Figure 3-1, inside a glove box under an argon-flooded atmosphere using a band saw. Subsequently, the sample pieces are subjected to a cascade washing in dimethyl carbonate (DMC) for at least 24 hours before evacuating at approximately 0.06 bar in the CitoVac (Struers) to vaporise residues of solvent and electrolyte. Afterwards, the samples are embedded in epoxy resin with the separating plane facing downwards to improve the stability of the microstructure. Then successively finer silicon carbide (SiC) abrasive papers are applied to remove material from the sample surface until the desired surface quality is achieved. First, the aim is to remove at least 2mm from the splitting plane to reduce any artefacts to a minimum (see Figure 3-2). Further preparation steps include gentle lapping, fine grinding and polishing with different lubricants and diamond suspensions. After polishing, various microstructural constituents can be easily visualized under the microscope.

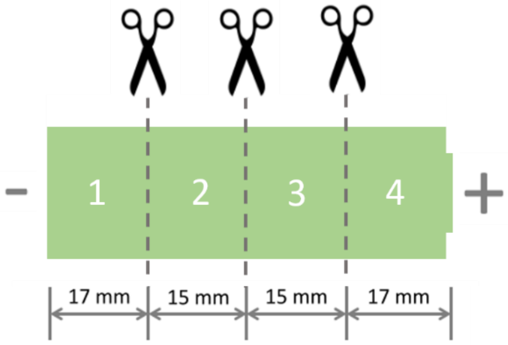


Figure 3-1: Schematic representation of a cell sample and its division into test pieces.

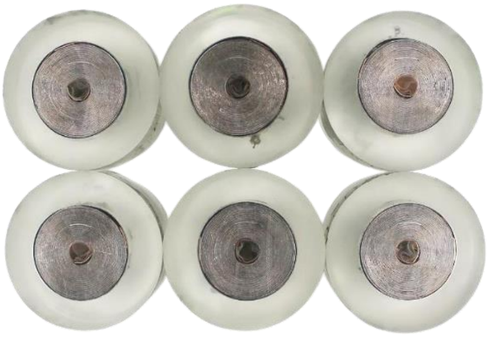


Figure 3-2: Shows cylindrical cross-sections obtained from three cells after the initial grinding step with the SiC abrasive paper.

3.3 Investigated sintered NdFeB magnet specimens

Similar to the experiments on Li-ion batteries, all investigations were performed on commercial NdFeB magnets. Specifically, for training and evaluating the unsupervised defect detection models, the magnets obtained for the investigations were produced by sintering using powder metallurgical processes as described in Section 2.2. Table 3-2 lists all the specimens used in our experiments and their magnetic properties.

Table 3-2: Listing of all the examined magnets and their properties. Note that the exact chemical composition of specimen 6 is not known.

Specimen	Manufacturer	Cross-section label	Remanence Br	Coercivity HcJ	Max. energy product
			T	kA/m	kJ/m^3
1	Manufacturer 1	HS17051_15	1.19	2532	273
2	Manufacturer 2	HS17051_22	1.14	2945	252
3	Manufacturer 1	HS17051_35	1.21	1080	281
4	Manufacturer 2	HS17051_90	1.14	2945	252
5	Manufacturer 3	HS17051_75	1.38	2046	369
6	Manufacturer 4	HS17051_03	1.34	1569	350

Specimen	Chemical composition [at %]									
	Fe	Nd	Dy	Cu	Co	Pr	Ga	Al	Ti	Gd
1	81.4	9.6	3.4	-	2.4	2.8	-	0.3	-	-
2	81.2	12.0	5.5	-	1.3	-	-	-	-	-
3	83.4	9.2	-	-	0.7	2.5	-	0.9	-	3.3
4	81.2	12.0	5.5	-	1.3	-	-	-	-	-
5	82.7	10.7	1.3	0.2	0.8	2.4	0.1	1.7	0.1	-

3.4 Sintered NdFeB magnet sample preparation

For the light microscope examination, cross-sections of the magnet samples were prepared based on standard metallographic procedure. First, the specimens were embedded using PolyFast (Struers), a conductive hot mounting medium suitable for fast mounting, in sections with an outer diameter of 30mm. Next, the grinding preparation is carried out in RotoPol-31 (Struers) up to a grit size of #1200, using SiC abrasive paper. This was followed by several polishing steps with a diamond suspension of 9 $\mu m,$ 3 $\mu m,$ 1 μm up to $\frac{1}{4}$ μm grit size. Examples of embedded cross-sections of the prepared magnet specimens suitable for light microscope examination are shown in Figure 3-3.

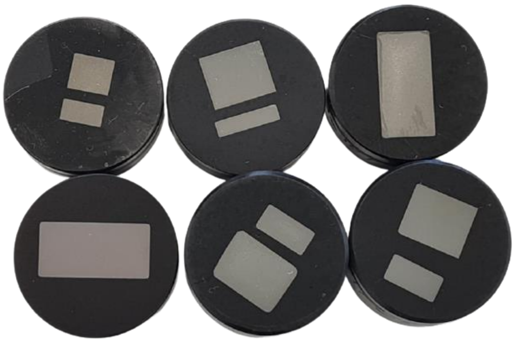
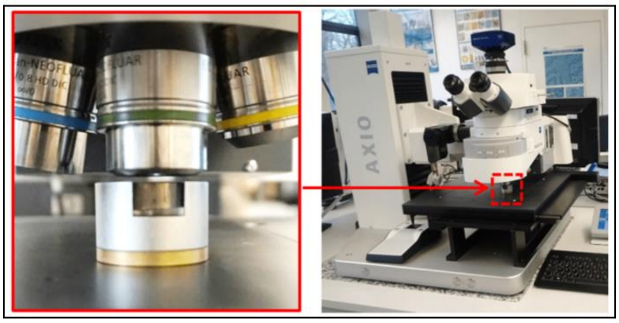
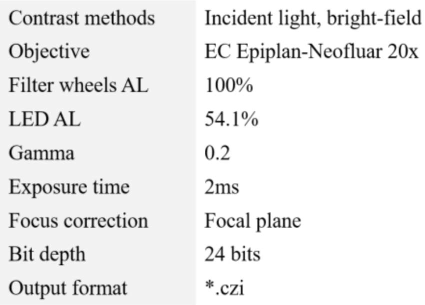


Figure 3-3: Cross-sections of prepared magnet specimens.

3.5 Light microscopy, image acquisition and processing

After achieving the desired sample grinding surface quality, high-resolution images of the relevant areas of the specimens are acquired in bright-field illumination using a light microscope of the type Carl Zeiss Axio Imager.Z2m Vario with the aid of the image analysis software Zen core 2.5 and the image acquisition settings listed in Figure 3-4. Once the acquisition is finished the complete image of the cross-section surface is saved in the software's czi format. The file size typically ranges from around 20 GB to over 100 GB depending on the size of the sample and bit depth. Subsequently, the large image files are converted to smaller patches better suited for image processing with open source deep learning software packages.





Contrast methods	Incident light, bright-field
Objective	EC Epiplan-Neofluar 20x
Filter wheels AL	100%
LED AL	54.1%
Gamma	0.2
Exposure time	2ms
Focus correction	Focal plane
Bit depth	24 bits
Output format	*.czi

Figure 3-4: Axio Imager.Z2m Vario light microscope (Zeiss) for automated image acquisition of prepared specimen (left). Acquisition parameters for digital imaging of the prepared specimen (right).

3.6 Software and hardware specification

Aside from the sample preparation and image acquisition steps described above, all subsequent image processing and computer vision experiments were carried out using python 3.6 and the scientific computing open-source libraries. Simple Big Image (SBI) software library [154] was used to convert large czi image files into png image patches. Apache MXNet v1.5.0 with the Gluon high-level framework was used to develop neural network models and experiment with various deep learning architectures. The models were trained on a Dell workstation with a 24 GB NVIDIA Quadro M6000 graphics processing unit (GPU).

3.7 Detecting defects in lithium-ion battery

Building a robust image classifier or object detection model involves several steps regarding the training data set selection, network architecture design, evaluation criteria, hyperparameters optimization etc. This section starts with an overview of the design steps required for developing such a model, and each step is discussed in more detail in subsequent sections. Our main objective is to build a tool for quality assessment of Li-ion battery electrodes, whereby a large cross-section micrograph is fed as input, and the output is an image gallery containing all the detected defects with their corresponding statistics. The problem is approached from a data-driven perspective, using state-of-the-art computer vision techniques (convolutional neural network) to extract and classify features from the images into the appropriate category. However, due to the limited number of examples of defects in our initial data set, in the first step, a sliding window classifier is used to scan the entire cross-section for defects. Note that this is a limited setup for defect detection because we can only detect defects that fall within the specific window size. Nonetheless, this is used to iteratively build up our data set in order to develop more complex approaches. Section 3.10 describes approaches to detect defects of various sizes not constrained by the window size and can perform near real-time analysis.

In summary, our approach for defect detection can be divided into three parts: in the first part, a limited amount of examples of images with defects and without defects are collected for training a binary classifier to distinguish both classes. Subsequently, the trained classifier is applied to several full cross-section images and manually sorted the results into five defect categories. However, not every class contained enough representative examples after sorting the defect into their various classes. Therefore, the new data set consisting of multiple defect classes is used in the second part to train a multiclass classification model to give us a better overview of typical defect types. Finally, an object detection model is developed using a more substantial and representative data set to improve the accuracy of the detection model and to characterize each type of defect by verifying the chemical composition through correlative analysis techniques.

Overview of the design steps

The process for developing the quality evaluation model can be divided into three stages:

- Sliding window binary image classifier
- Multiclass sliding window classifier
- Real-time object detection model

Each stage of the model development process involves multiple iterations of the steps illustrated in Figure 3-5. The steps can be summarized as follows: first, a small amount of data is collected and labelled to train a model. Next, the data is cleaned to remove duplicates or bad images. Then the model is trained on the data set, and the best version based on the evaluation criteria is selected. Subsequently, the trained model is applied to the entire overview image of the battery; the result is analyzed and used to generate more labelled examples. The new examples are added to the initial data set, and the process is repeated several times until we achieve satisfactory performance.

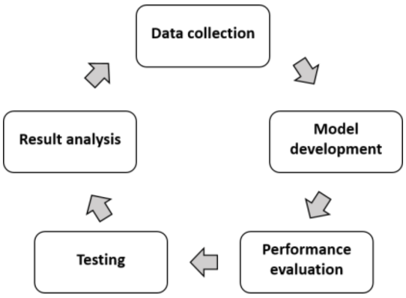


Figure 3-5: Illustrates the steps involved in developing the defect detection model.

3.8 Sliding window binary image classifier

A simple approach for localizing objects in an image is to slide a classifier over the image, independently classifying each image patch as being an object of interest or not. This typically involves four simple steps: scanning every location in the image with fixed-sized sliding windows, extracting features from each window position, and feeding the features into a classifier (e.g. SVM or MLP). Since objects can be of very different scales, the image is scanned at different scales forming an image pyramid. Lastly, non-maximum suppression (NMS) is used to ignore redundant, overlapping bounding boxes. A similar sliding window technique is developed for our first approach to detect defects in the electrodes of LIBs. Although more modern object detection techniques have superseded this approach, it is adopted to simplify the processing pipeline and keep the computational complexity low while building out the data set. The structure of the proposed framework is illustrated in Figure 3-6. For the training phase, patches of equal sizes are manually extracted from large micrographs of LIB electrodes and sorted into two categories, defect and no-defect, to create the initial training dataset. Once the model is trained, the performance is evaluated using a sliding window to scan the entire micrograph for new defects. This way, the effectiveness of the model can be quickly tested, and a larger data set containing more examples of defects can be collected within a reasonable amount of time and effort.

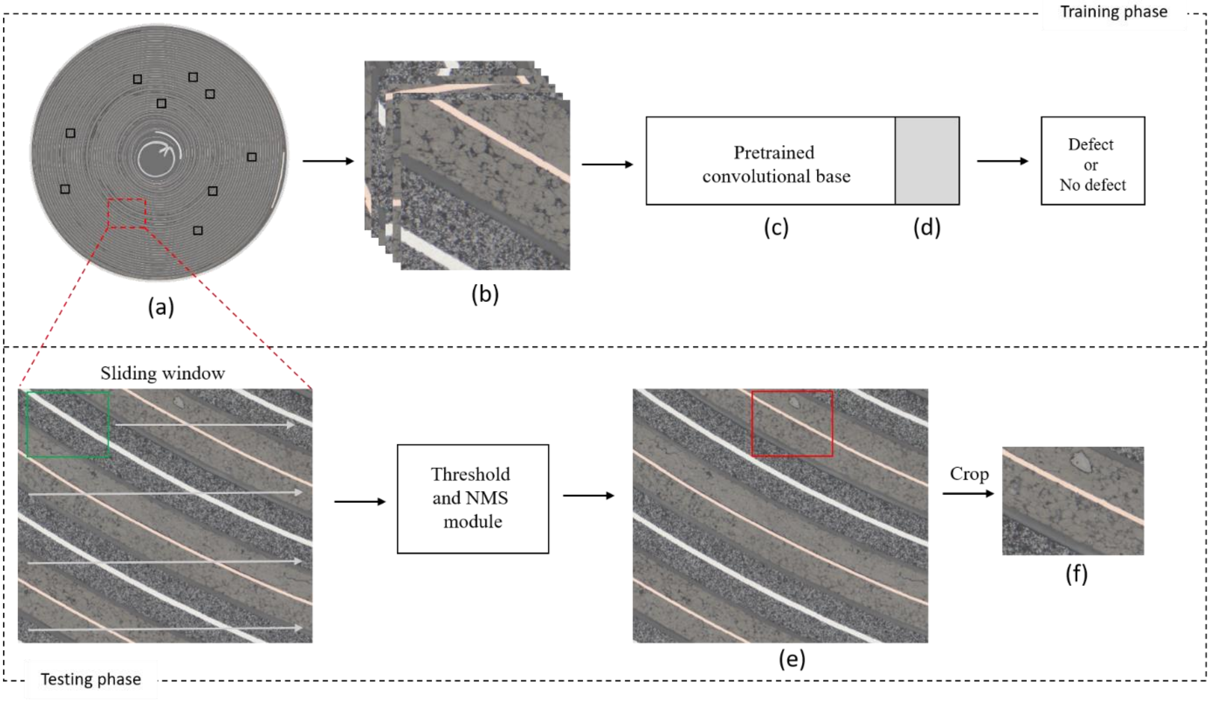
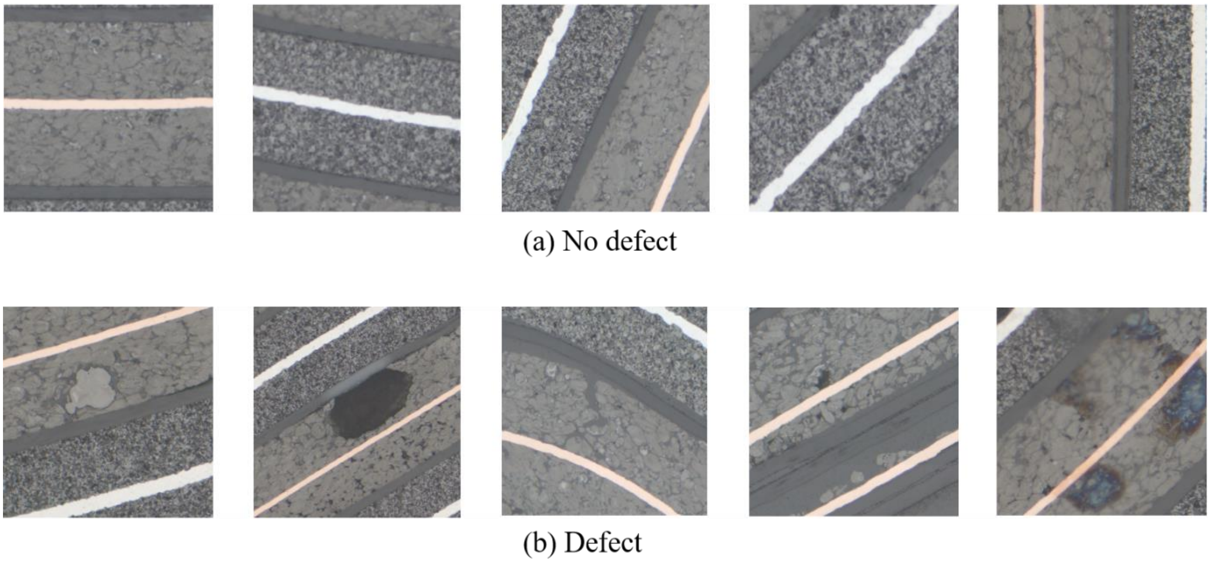


Figure 3-6: Structure of the proposed framework. (a) Large overview image of electrode micrograph. (b) Extracted patches. (c) Pretrained CNN. (d) New classification layer. (e) Defect detection and localisation. (f) Cropped output.

3.8.1 Dataset for training the binary classification networks

The initial dataset (Dataset-A) used for training the sliding window classifier contains 1540 RGB colour images of electrode patches. The images were extracted from four cross-section images of two batteries with the same chemistry (INSP_026 and INSP_033). Subsequently, the images were labelled and sorted into two categories; images without defects (1220) and images containing microstructural defects (320), such as foreign particle inclusion, pores, cracks and inhomogeneous coating. Examples of images from both classes are shown in Figure 3-7. One of the main challenges for training a supervised model for detecting defects is the large class imbalance between the number of examples without defects and examples with defects. Moreover, training on an imbalanced data set can bias the model towards the majority class. In our case, due to the infrequency of defects (~1 in $1\ \mathrm{mm}^2$) in the electrode samples, the data from the collection process are usually imbalanced. Obtaining examples without microstructural defects was much easier than the effort required to collect sufficient examples with defects. Also, because many of the defects have similar morphology to the electrode active material, this makes the problem even more challenging for manual visual inspection and a machine learning model. Nevertheless, in this classification task detecting rare defects while minimizing the number of false positives is essential.



(a) No defect

(b) Defect

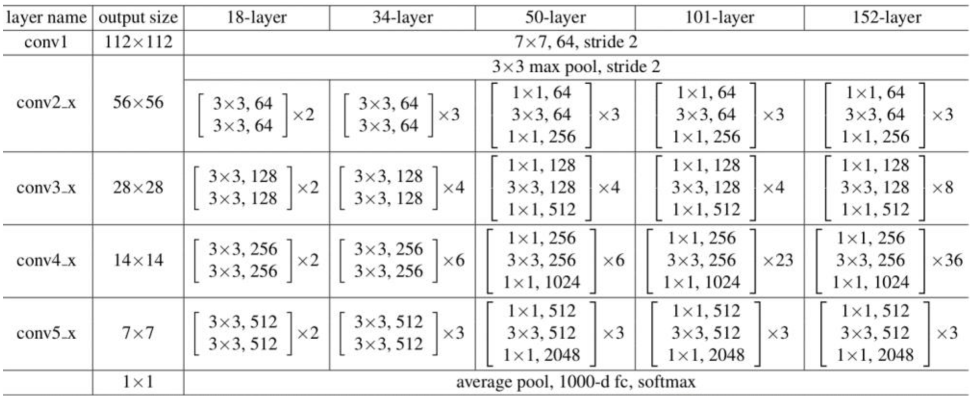
Figure 3-7: Examples of images from the binary classification data set. (a) Shows some examples of images that do not contain any defects. (b) Show examples of images containing at least one type of electrode defect.

3.8.2 Classification network architectures

In the past decade, many deep CNN architectures have been developed and evaluated on the ImageNet dataset and achieved outstanding results. A very effective and commonly used approach in deep learning to achieve good classification performance on relatively small datasets is to modify a network pretrained on a large dataset such as ImageNet and fine-tune the weights on the smaller dataset. Therefore, ResNet [126] and VGG [152] models were selected because they are one of the top-performing models on the ImageNet dataset and are also commonly used for many transfer learning and defect detection problems. In addition, their architectures are relatively simple compared to some other state-of-the-art models, which reduces the chance of the network overfitting to our much smaller imbalanced dataset. The performances of both models are evaluated using a standard cross-entropy loss function that gives equal importance to each class and to specifically designed loss functions and techniques for handling imbalanced data.

Residual Network (ResNet)

Previously, when training very deep neural networks using gradient-based learning and backpropagation, the gradient became very small to the point where it vanishes after a certain depth. Thus effectively preventing the weights from changing values and ultimately stopping the network from learning. This is commonly known as the vanishing gradient problem. The ResNet architecture was designed to prevent many of the issues that affected very deep neural networks, including the vanishing gradient problem, predominantly through residual connections, which allows the model to easily pass useful information from a previous layer to the next. There are several variants of the ResNet architecture named according to the number of layers, such as ResNet-18, ResNet-50, ResNet-101 and ResNet-152. We selected the smaller ResNet-50 model with 50 layers for our experiments because of our relatively small dataset. The ResNet-50 architecture is shown in Figure 3-8.

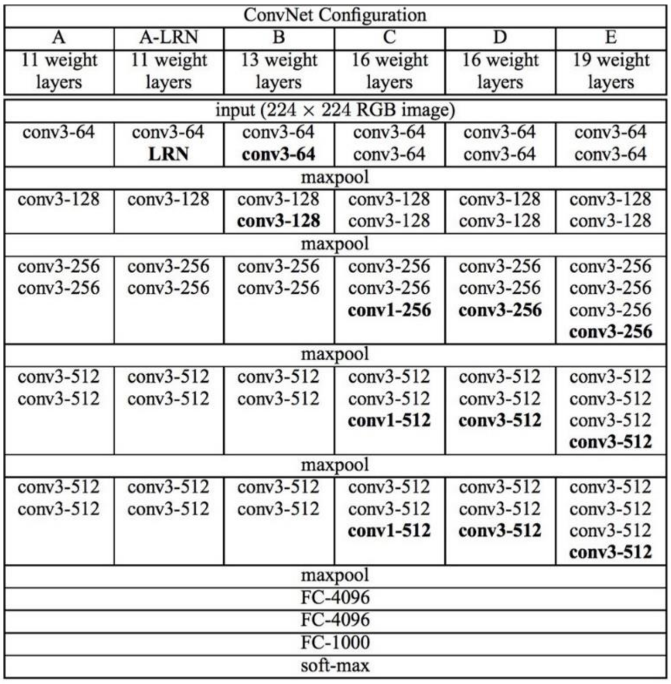


layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×8
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23	[1×1, 256 3×3, 256 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				

Figure 3-8: Shows various ResNet model architectures [126].

Visual Geometry Group (VGG)

The VGG architecture has been one of the most widely used models in computer vision over the last few years. It mainly consists of stacked convolutional and max pooling layers. The VGG model improves over prior state-of-the-art models by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layers) with multiple 3x3 kernel-size filters. The main idea is that using a smaller receptive field is better than using a large filter size because multiple non-linear layers can be incorporated, allowing the network depth to be increased and more complex features to be learned while decreasing the number of parameters. In addition, the representations learnt by the model are known to generalise well to other datasets. There are two common variants of the VGG model, namely, VGG-16 and VGG-19, which have 16 and 19 layers, respectively. The VGG-19 architecture was selected for our experiments since it achieved the best performance in one of our previous studies on a similar data set [57]. The VGG-19 architecture is shown in Figure 3-9.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3-9: VGG model architectures. The convolutional layer parameters are denoted as “conv(receptive field size)-(number of channels)” [152].

3.8.3 Evaluation metrics and loss functions for the image classifiers

We evaluate the predictive performance of each method against the ground truth labels using the following standard metrics:

- Accuracy
- F_1-score
- Area under receiver operating characteristic curve (AUROC)
- Area under precision recall curve (AUPRC)
- Confusion matrix

Accuracy. Classification accuracy is the most frequently used metric to evaluate the performance of a learning system on classification problems. Accuracy is calculated as the number of examples in the test set that were correctly predicted, divided by the total number of samples in the test set.

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{Total Predictions}} \tag{3.1}$$

However, when the dataset is severely skewed, accuracy can become an unreliable measure of model performance because the impact of the least represented but more important examples is reduced compared to the majority class. For instance, considering a data set with a 100:1 class imbalance, 99% classification accuracy can be achieved by simply predicting the majority class for all examples in the test set, which is clearly undesired and can lead to erroneous conclusions. Therefore, the model performance is also evaluated using F_1-score and AUROC, which are frequently used measures for imbalanced data problems.

F_1-score. The F_1-score is a metric to evaluate the accuracy of the predictions. It combines as a weighted average (harmonic mean) both the *precision* and *recall* to compute the score, featuring a score of 1 as the best score and 0 as the worst score. The relative contribution of precision and recall to the F_1-score are equal.

$$\text{F}_1\text{score} = \frac{2 \text{ x precision x recall}}{\text{precision} + \text{recall}} \tag{3.2}$$

Precision is the fraction of true positive (TP) results among the true and false positive (TP + FP) results returned by the classifier.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.3}$$

Recall is the fraction of true positive (TP) results among all the samples that should have been identified as positive (TP + FN). Note that we do not use the true negatives when calculating precision and recall. They are only concerned with the correct predictions for each class. For a good model, both precision and recall should be high.

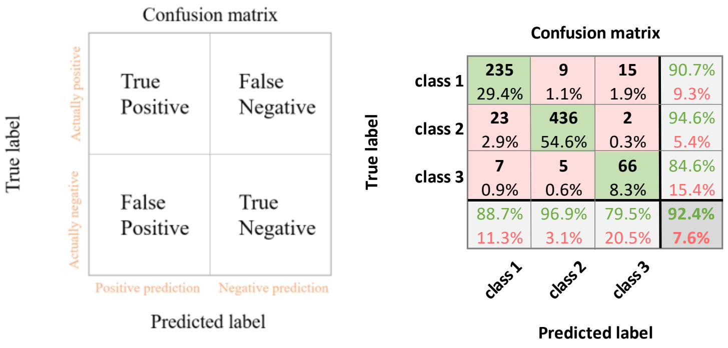
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.4}$$

AUROC. The receiver operating characteristic (ROC) curve is a plot of the false positive rate (FPR) on the x-axis and the true positive rate (TPR) on the y-axis for various thresholds between 0 and 1. The area under the curve (AUROC) can be used to summarize the model's predictive performance, where a larger area under the curve is often better. In the top left corner of the plot, a false positive rate of zero and a true positive rate of one is the “ideal” point. Likewise, since we would like to maximize the true positive rate while minimizing the false positive rate, the ‘steepness’ of the ROC curve is also important. AUROC values close to 1 are considered good, while AUROC values close to 0 imply the model is bad.

$$TPR = \frac{TP}{TP + FN}, \qquad FPR = \frac{FP}{TN + FP} \tag{3.5}$$

AUPRC. The model outputs a probability distribution for each class, and a threshold of around 0.5 is typically chosen to make a prediction, which is not always ideal. Depending on the selected threshold, the value of precision and recall can change significantly. Therefore, it is possible to create a plot between these sets of values by calculating the precision and recall values for various threshold values. This curve is known as the precision-recall curve, and the area under the curve is the AUPRC. Both AUPRC and AUROC are useful metrics because the curves of different models can be directly compared. However, AUPRC is preferred when there is a large class imbalance in the data set [155].

Confusion matrix. The confusion matrix C is a specific table layout that allows for easy visualization of the performance of a machine learning algorithm. Each row of the matrix corresponds to instances in the true class, while each column represents instances in the predicted class. In other words, entry $C_{i,j}$ is equal to the number of observations which should be in group i and predicted to be in group j. An example of a confusion matrix is shown in Figure 3-10. The confusion matrix should only be filled diagonally from left to right when all examples are correctly predicted.



	class 1	class 2	class 3	
class 1	235 29.4%	9 1.1%	15 1.9%	90.7% 9.3%
class 2	23 2.9%	436 54.6%	2 0.3%	94.6% 5.4%
class 3	7 0.9%	5 0.6%	66 8.3%	84.6% 15.4%
	88.7% 11.3%	96.9% 3.1%	79.5% 20.5%	92.4% 7.6%

Figure 3-10: Example of a confusion matrix for three classes (right). The last row and the last column (light grey) show each class's precision and recall, respectively.

Weighted cross-entropy. The weighted cross-entropy (WCE) is a variant of the cross-entropy loss function where all positive examples are weighted by some coefficient such that poor predictions of the minority class are penalised more heavily. It is typically used when there is a class imbalance. For instance, if the majority to minority class ratio is 100:1, the regular cross-entropy does not compensate for this imbalance and could bias the model towards the majority class. WCE can be defined as follows:

$$WCE(p, \hat{p}) = -(\alpha p \log(\hat{p}) + (1-p)\log(1-\hat{p})) \tag{3.6}$$

where $\hat{p}$ is the prediction probability of the model, and p is the ground truth label. α is the weight term used to balance the importance of positive and negative examples. A value of $\alpha > 1$ is used to decrease the number of false negatives, and $\alpha < 1$ is used to decrease the number of false positives.

Focal loss. While α balances the importance of positive and negative examples, it does not differentiate between easy and hard examples. The focal loss (FL) tries to compensate for this by adding a modulating factor $(1-\hat{p})^{\gamma}$ to the cross-entropy loss. Thus FL can be defined as follows:

$$FL(p,\hat{p}) = -(\alpha(1-\hat{p})^{\gamma} p \log(\hat{p}) + (1-\alpha)\hat{p}^{\gamma}(1-p)\log(1-\hat{p})) \tag{3.7}$$

where $\gamma \geq 0$ is a focusing parameter that reshapes the loss function to down-weight the contribution of easy examples so that the model focuses training on the hard examples. When $\gamma = 0$, this gives the balanced cross-entropy function. The hard examples are those samples the model misclassifies with a high probability, thereby producing large error signals. When an example is correctly classified with a high probability (ground truth label is 1 and $\hat{p} \rightarrow 1$), the value of $(1-\hat{p})$ is small. When this term is raised to the power of γ, the modulating factor goes to 0, and the loss for easily classified examples is down-weighted, effectively reducing the loss contribution from easy examples. On the other hand, if an example is misclassified with low probability, the modulating factor is close to 1, and the cross-entropy loss is unaffected.

3.8.4 Experiment settings for the binary classifier models

First, the data set is randomly split such that 80% is used for training and the remaining 20% is used for testing. The training set consists of 1232 images, of which 976 have no defects, and 256 contain defects. The test set consists of 308 images, 244 images with no defects and 64 images with defects. During training, the images are resized to 256 x 256 pixels, and the pixel values are rescaled in the range [0, 1]. Five-fold cross-validation is used to evaluate the settings and tune the hyperparameters of the models on the training set. The model weights are initialized with pretrained weights from the ImageNet dataset and fine-tuned using our training set. The image pixels are normalized by subtracting the mean (0.485, 0.456, 0.406) and dividing by the standard deviation (0.229, 0.224, 0.225) of the ImageNet dataset. The batch size is set to 16 and trained for 60 epochs. The network is trained using Nesterov accelerated gradient (NAG) descent optimizer with the initial learning rate set to 0.001 for the first 20 epochs, which we subsequently decay after every 20 epochs by a factor of 0.75. The momentum and weight decay parameters were set to 0.9 and 0.0001, respectively.

During training, various data augmentation techniques are used to prevent overfitting, such as randomly flipping the image both horizontally and vertically, colour jittering, and randomly changing the image intensity. For the focal loss function, there are two hyperparameters, α and γ, namely. Various values for α in the range of [1, 2, 3, 4] and γ in the range [1, 2, 5] were evaluated. After obtaining the optimal set of parameters, the model was trained on all samples in the training set and evaluated on the test set. In addition, a model trained with focal loss function is compared to one trained with a class weighted cross-entropy loss function. For the weighted cross-entropy loss function, the balancing term α is set to 4 since the ratio of the majority class to the minority class is 4:1. Furthermore, over-sampling and under-sampling techniques are used to balance the majority and minority class ratio. For the over-sampling approach, defects from various positions and zoom levels are cropped from the original cross-section image to artificially increase the number of samples in the minority class to equal the number of samples in the majority class. Two examples of the same defects from this data set are shown in Figure 3-11. For the under-sampling approach, samples are randomly removed from the majority class until the number of samples in the training set for the majority and minority classes are equal.

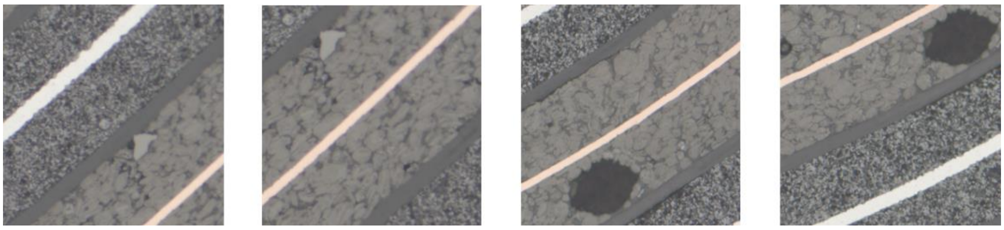


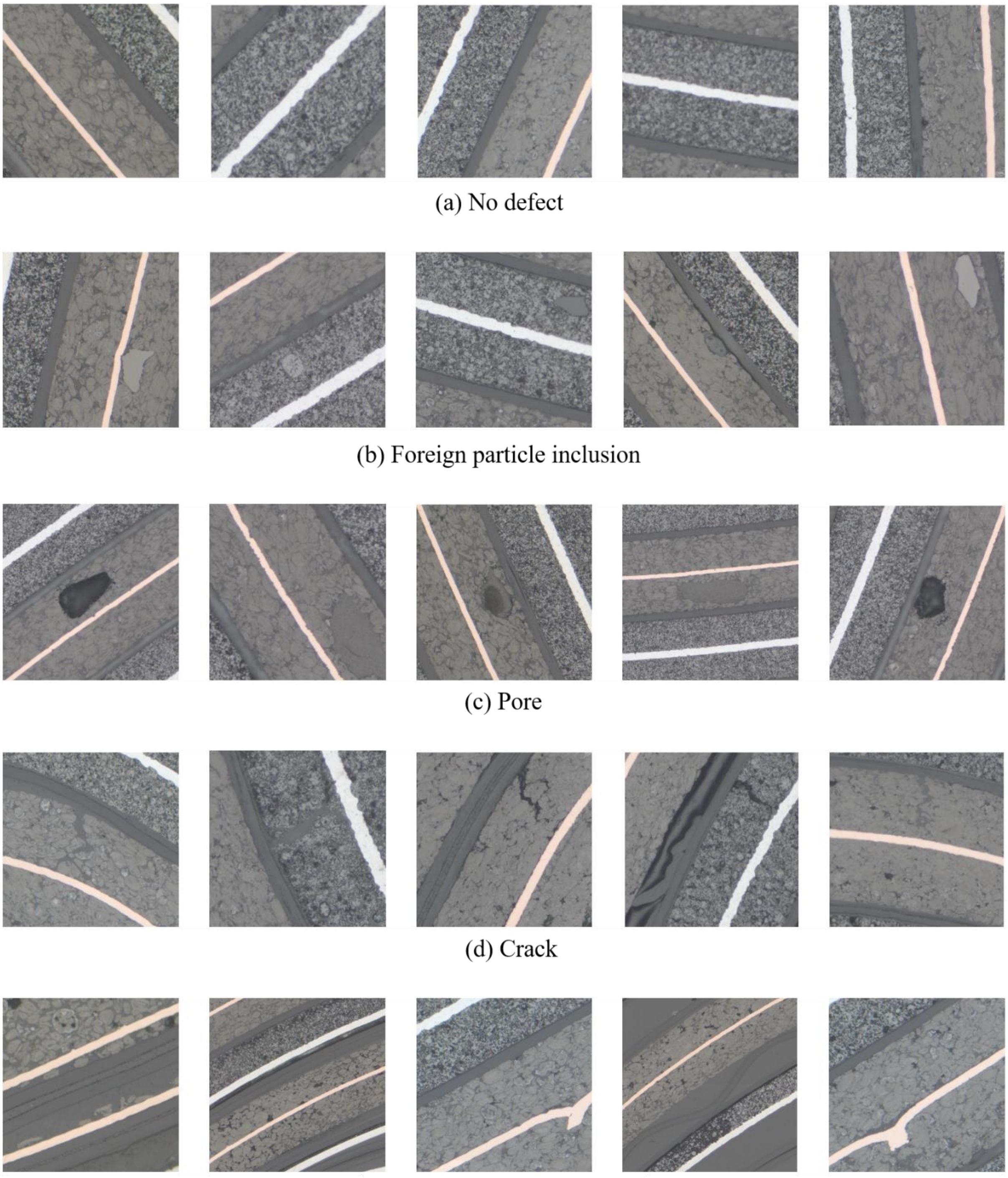
Figure 3-11: Shows two examples of the same defect cropped from different positions to artificially increase the size of the dataset for oversampling.

3.9 Sliding window multiclass image classifier

In this section, further improvements to the defect detection model are made by collecting more examples of defects using the networks developed in the previous section and manually labelling and sorting the results into five main categories: foreign particle, crack, pore, geometric irregularity and preparation artefact. Including the examples with no defects, there are now six classes of images in total. Subsequently, both classification models, ResNet-50 and VGG-19, are retrained to classify image patches into one of the six categories.

3.9.1 Dataset for training the multiclass classification networks

The improved dataset (Dataset-B) contains 4908 image patches collected from 15 cross-section images of 5 battery samples with similar chemistry (INSP_018, INSP_026, INSP_030, INSP_033 and INSP_038). The categories of defects are distributed as follows: 1311 images with no defects, 749 images with foreign particle inclusions, 362 images with pores, 334 images with cracks in the active materials, 529 images with geometric abnormalities and 1623 images with preparation artefacts. Any object which does not have the same appearance as either the cathode or anode active material is categorized as a foreign particle. Pores are small holes or void spaces not occupied by the main active material. However, in some cases, the pores can be filled during the sample preparation process, making it inconspicuous and more difficult to detect. The cracks are small splits in the active material, mostly occurring at the corner of the inner windings of the electrode layers. Non-uniform electrode coating, a separation between the anode and cathode layer and deformation of the current collector are categorized as geometric irregularities. Other defects, including breakout of cathode active material, delamination, water and ethanol contamination, are categorized as preparation artefacts. Some representative examples of images from each defect category are shown in Figure 3-12.



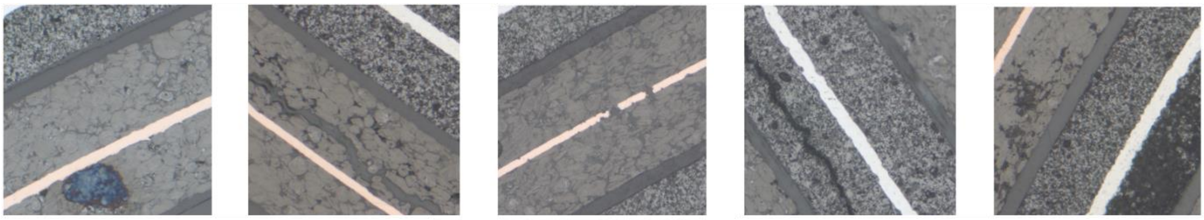
(a) No defect

(b) Foreign particle inclusion

(c) Pore

(d) Crack

(e) Geometric irregularity



(f) Preparation artefact

Figure 3-12: The five categories of electrode defects considered in this experiment.

3.9.2 Experiment settings for the multiclass classifier models

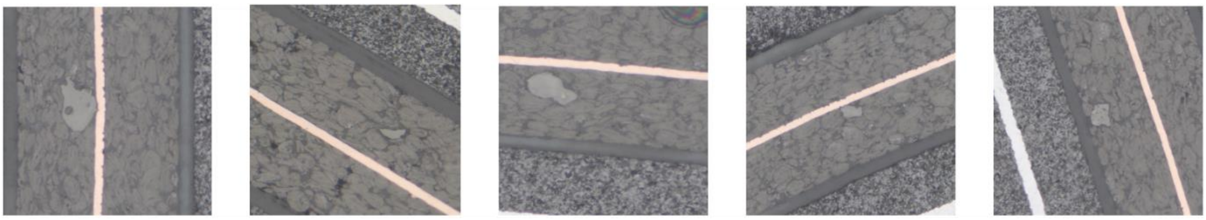
As described in the previous experiment, the dataset is randomly split using 80% for training and the remaining 20% for testing. This dataset contains 3926 images in the training set and 982 images in the test set. All the images are resized to 256 x 256 pixels during training, and the pixel values are rescaled in the range [0, 1]. The weights of the convolutional layers were initialized with the pretrained weights of the best model from the previous dataset (Dataset-A), and the output layer weights were initialized using the Xavier initializer [156] before fine-tuning all the layers using the new training set. The image pixels are normalized by subtracting the mean and dividing by the standard deviation of the ImageNet dataset. The network is trained using a batch size of 16 samples for a maximum of 60 epochs with early stopping criteria and Nesterov accelerated gradient (NAG) descent optimizer. The initial learning rate is set to 0.001 for the first 20 epochs, decaying by a factor of 0.75 after every 20 epochs. The momentum and weight-decay parameter were set to 0.9 and 0.0001, respectively. Various data augmentation techniques such as randomly flipping the image horizontally and vertically, randomly rotating the image between -180 and 180°, colour jittering, and randomly changing the image intensity were also applied during training to prevent overfitting. Values for α in the set [1, 2, 3, 4] and γ in the set [1, 2, 5] were evaluated. For the weighted cross-entropy loss function, weights of $\alpha = [4, 3, 1, 2, 4, 1]$ were applied for the crack, geometric irregularity, no defect, foreign particle, pore and preparation artefacts, respectively, based on the sample size distribution. Lastly, the over-sampling technique's effect on balancing the majority and minority class ratio was investigated. We did not investigate the effect of the under-sampling technique in this experiment as it did not improve the performance of the binary classifier.

3.10 Automatic localization of electrode defects using CNN

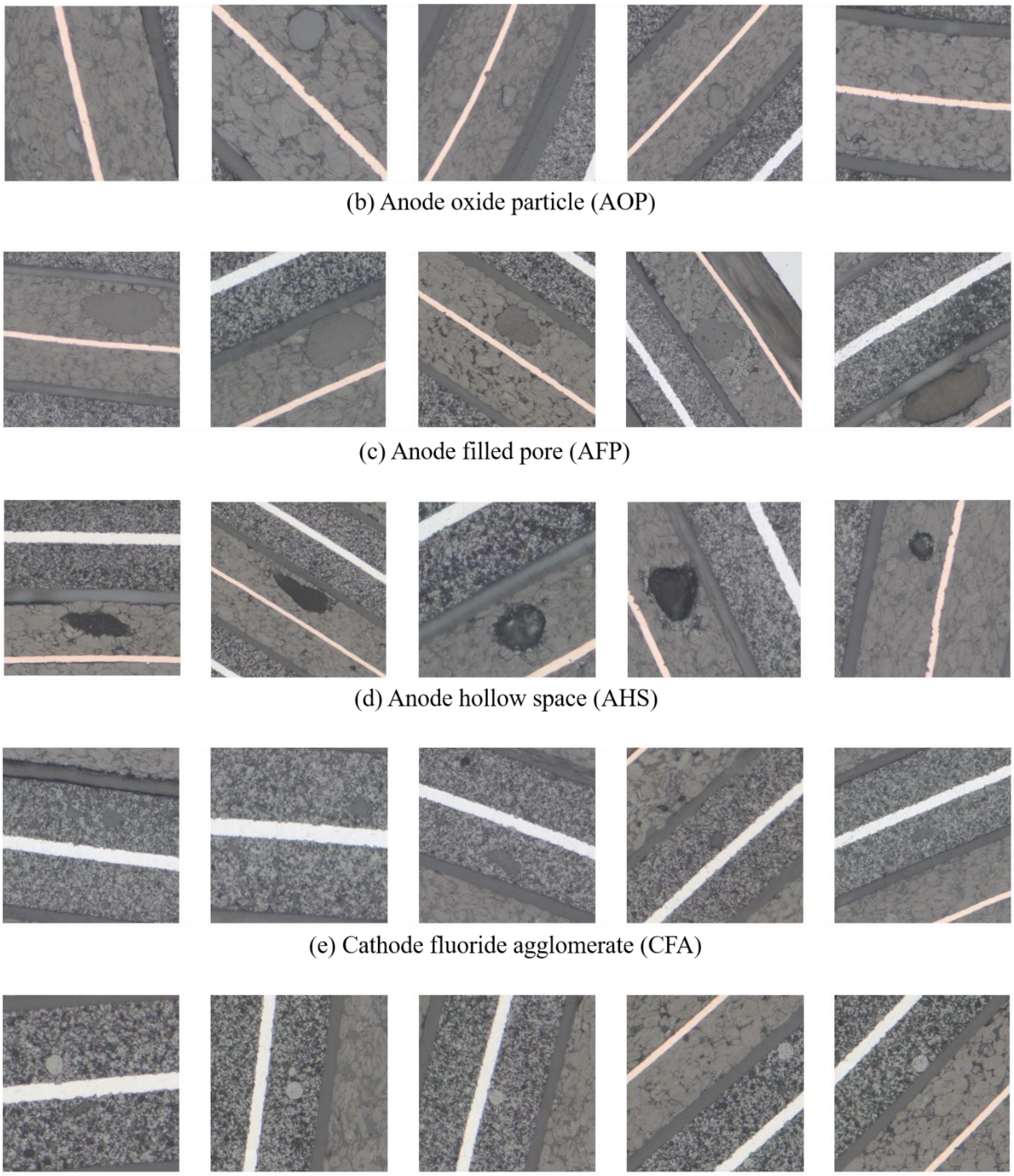
The following sections develop defect localisation models that overcome the limitations of approaches for detecting electrode defects described in Sections 3.8 and 3.9. Our main approach is based on recent successful CNN-based object detection models for feature extraction and object localization. While many neural network architectures have been proposed for object detection, there is still no consensus on the best method for any given task. Therefore, the primary focus here is investigating the effectiveness of the two-stage detector Faster R-CNN and one-stage detector SSD described in Section 2.7.3 and Section 2.7.4, respectively, for the defect detection task. Though these architectures were originally presented with a particular feature extractor module, both the ResNet and VGG network are investigated as the backbone feature extractor network since they effectively extracted useful features for classifying the electrode defects. These feature extraction modules are responsible for transforming raw image pixels of the input image into high-level featurized representation in order to generate tight-fitting bounding-box coordinates around each detected defect in the image along with their respective class labels. In the next section, the dataset used in the experiments to evaluate the effectiveness of the object detection models is described, followed by a description of the evaluation criteria used to judge the performance of each model and then the specific implementation details of each model architecture. Afterwards, the training procedure and the hyperparameter settings used in the experiments are presented. The performances of the models are evaluated in Section 4.1.4.

3.10.1 Dataset for training the defect localization networks

The sliding window multiclass classification model developed in the previous section was applied to full cross-section images to discover patterns in the detection results and build up our data set. The detected defects were manually categorized based on whether they occurred in the anode or the cathode. This results in 10 classes of defects, namely: anode carbon particle (ACP), anode oxide particle (AOP), anode filled pore (AFP), anode hollow space (AHS), cathode fluoride agglomerate (CFA), cathode round particle (CRP), cathode oxide particle (COP), geometry, crack, and preparation artefacts. The complete dataset consists of 3296 RGB colour images. The distribution of defects present in the dataset is shown in Figure 3-19. Defects are named based on where they occur in the electrode, the main component present in the defect if known and the type of defect. For instance, cathode fluoride agglomerate (CFA) refers to an agglomerate with a high fluoride content present in the cathode. Subsequently, the images are annotated with tight-fitting bounding boxes using the labelling software LabelImg, which saves the category and coordinates of each defect in corresponding XML files. Figure 3-13 shows some examples of defects from the new categories, and Figure 3-15 presents some annotated examples used to train the defect detection and localisation model.



(a) Anode carbon particle (ACP)



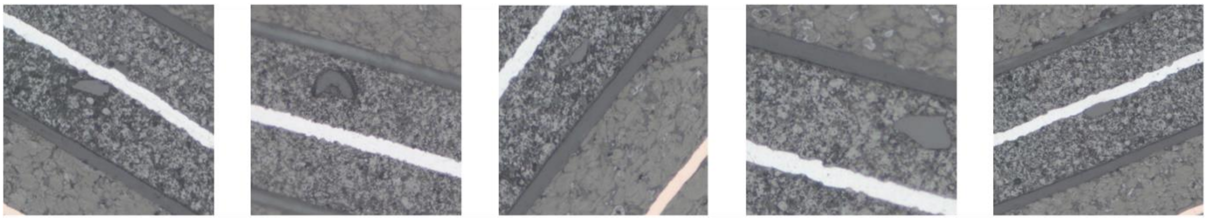
(b) Anode oxide particle (AOP)

(c) Anode filled pore (AFP)

(d) Anode hollow space (AHS)

(e) Cathode fluoride agglomerate (CFA)

(f) Cathode round particle (CRP)



(g) Cathode oxide particle (COP)

Figure 3-13: Examples of the new categories of defects included in the dataset.

3.10.2 Evaluation metrics for the defects localization networks

Different evaluation criteria are required for defect detection and localisation than for image classification since we are not just predicting the class of a single defect in an image but now want to detect multiple defects in the same image and their respective locations. As a result, the performance of the model is typically evaluated using the following metrics:

Intersection over Union (IoU). The IoU is used as a standard metric to determine whether a bounding box prediction is considered a true positive. A detection is considered a true positive if the IoU with the ground-truth box is greater than a defined threshold. The true positive in this case refers to the amount of correctly predicted ground-truth boxes; false positive is the number of non-defects that are wrongly predicted as defects, and false negative refers to the number of real defects which are missed. In object detection, true negative does not exist. Every detected box is considered a true positive or false positive by evaluating the area of overlap between the ground truth and the predicted box. If the area of overlap a_o is above the defined threshold, typically 0.5, the predicted box is regarded as true positive. The IoU is calculated as follows:

$$a_o = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{3.8}$$

where $B_p \cap B_{gt}$ denotes the intersection of the predicted bounding boxes and ground truth bounding boxes, while $B_p \cup B_{gt}$ denotes their union, as illustrated in Figure 3-14.

Average precision (AP). The average precision for each class is calculated by computing the area under the precision-recall curve. First, divide the recall value from 0 to 1.0 into 11 points – 0, 0.1, 0.2 …1.0, then compute the average of the maximum precision value for these 11 recall values. This is defined as:

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,0.2,\ldots,1.0\}} \max_{\tilde{r} \geq r} p(\tilde{r}) \tag{3.9}$$

where $p(r)$ is the measured precision at recall r.

Mean average precision (mAP). Considering that there are target objects from different classes in the dataset, first compute the average precision for each class separately, then average over the number of classes to obtain the mean average precision. This value allows us to compare the robustness of different models for all classes easily and is defined as follows:

$$mAP = \frac{1}{N_{cls}} \sum_{i} AP_i \tag{3.10}$$

where N_{cls} is the number of classes and AP_i is the average precision value for class i.

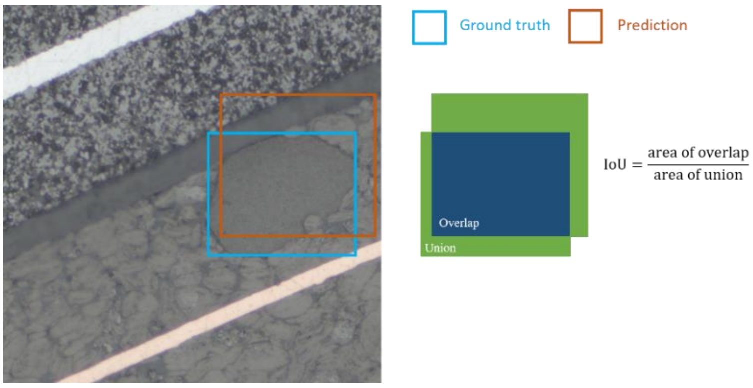


Figure 3-14: Illustration of the intersection over union metric used to train the defect localization models.

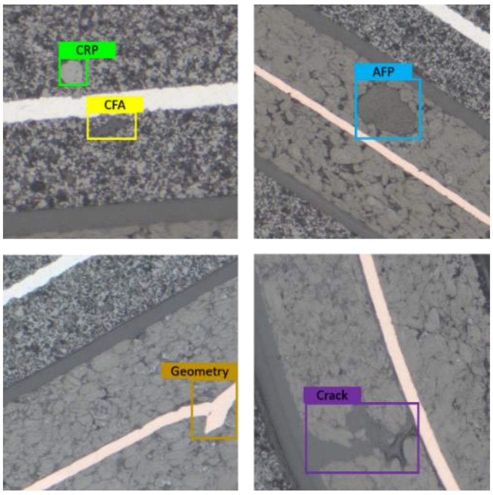


Figure 3-15: Examples of tight-fitting ground-truth bounding box labels.

3.10.3 Implementation details of the defect localization networks

Faster R-CNN based detection system

As described in Section 2.7.3, Faster R-CNN is an integrated method that generates regional proposals using the Region Proposal Network (RPN) and then performs object detection using a shared convolutional base for feature extraction. This section describes the implementation details used to design the Faster R-CNN model architecture for defect detection. We compared the VGG-19 architecture to the ResNet-50 architecture used for image classification in the previous sections for the feature extraction module. Features are extracted from the last layer of the "conv4" block of the ResNet-50 network and from the last layer of the fourth block of the VGG-19 network and used as input to the RPN. In order to generate region proposals, the RPN convolves a CNN with kernel size 3×3, padding 1, stride 1 with 1024 output units on the output feature map of the feature extraction network. This produces a 1024-d feature map for every anchor box, which is fed into two sibling layers: a 1×1 convolution layer with $4k$ output units for bounding box regression and a 1×1 convolution layer with $2k$ output units for object classification.

For instance, given an input image of 600×800 pixels and passing it through the feature extraction module, after a subsampling ratio of 16, the output will have a dimension of $[1024 \times 38 \times 50]$. It is common to use anchor boxes with areas of $(128^2, 256^2, 512^2)$ pixels and aspect ratios of (1:1, 1:2 and 2:1) to detect common objects like tables and cars. However, many of the defects in our dataset are on the scale of 25×25 pixels or smaller. Therefore, in this work, anchors are generated with 5 scales (2, 4, 8, 16, 32) and 3 aspect ratios (0.5, 1, 2) in order to match the size and scale of objects in our dataset and detect defects of very different scales and aspect ratios, with the smallest anchor box chosen to be 16×16 pixels, thus yielding a total of $k = 15$ anchors at each sliding position and 28500 $(38 \times 50 \times 15)$ anchors in total. Then the RPN is applied on the output feature map, which generates proposals with output dimension (28500×4) for bounding box regression and their respective objectness score with dimension (28500×2) for object classification, as shown in Figure 3-16. As a result, the RPN produces proposals for each anchor box and their objectness scores. An anchor box is assigned a positive label if it has the highest IoU overlap with a ground truth label or IoU greater than 0.7 with any ground truth label. A non-positive anchor box is assigned a negative label (background) if it has IoU less than 0.3 for all ground truth boxes. All the other anchor boxes with IoU values between 0.3 and 0.7 and anchor boxes that fall outside the image are ignored and do not contribute to the training objective. Since a much larger ratio of the generated anchor boxes will have negative labels, we randomly sample 128 positive anchors and 128 negative anchors, giving a total of 256 anchors for training using the smooth L1 loss for bounding box regression and cross-entropy loss for classification, as shown in Equation (2.45). The regression outputs are offset with anchor box locations using Equation (2.47).

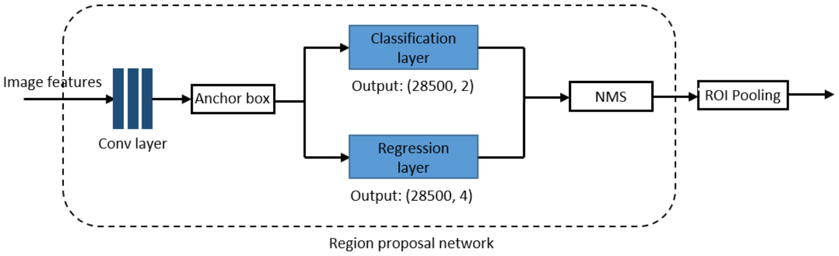


Figure 3-16: The Region proposal network takes a feature map as input and outputs a set of rectangular object proposals, each having a score describing whether the region contains an object or not.

Once the RPN outputs are generated, before sending to the ROI pooling layer for extracting fixed-length feature vector, the region proposals need to be pre-processed since there is a high degree of overlap between the generated bounding box proposals. Therefore to reduce redundancy, non-maximum suppression (NMS) is applied to the region proposals based on their class scores, using an overlapping threshold of 0.7 to define the minimum area required to remove overlapping bounding boxes. This results in about 2000 region proposals per image, and we select the top-N ranked proposals for detection (2000 proposals while training and 300 proposals while testing). This gives the final region proposals used as input to the Fast R-CNN detector, which tries to predict the locations of defects for the proposed boxes and the corresponding class label for each proposal. Note that since each mini-batch is collected from a single image containing many positive and negative proposals, but the vast majority of samples will be negative and not have any defects, there will be an extreme class imbalance while training that could bias the network towards negative samples. Therefore, a random sample of 128 boxes from the top 2000 region proposals is taken to form a mini-batch, and the losses are computed only for these boxes. A positive ratio of 0.25 is used to select the number of positive examples out of the 128 samples. If there are more than 32 positive samples in an image, 32 proposals are sampled from the positive ones, and if there are fewer positive boxes, the samples are padded with negative boxes. For a region proposal to be positive, the minimum overlap with any ground truth label is set to 0.5. Next, ROI pooling is used to extract features from the output of the base feature extractor network for each of the sampled regions proposed by the RPN. Note that the input image is passed only once, and the computed features are shared across all the region proposals. Considering that we have objects of various sizes in the input image, the ROI pooling layer takes a section of the output feature map corresponding to the region proposal and scales it to a pre-defined size of 7×7. This is done by dividing the region proposal into sections with the same dimension as the desired output and computing the largest value in each sub-window using the max pooling operation. Therefore, we get consistent feature maps with fixed sizes from proposals of different sizes, which saves a lot of processing time since the same input feature map for all the regions can be used. Each feature vector from the ROI pooling layer is fed into a sequence of fully connected layers that branch into a classification and regression head, as shown in Figure 3-17. The output vector from the classification head contains probability estimates for each of the K defect classes plus an additional background class, while the regression head outputs refined bounding-box positions for one of the K classes. Finally, the two sibling output layers are used to compute the multi-task loss on each labelled ROI using Equation (2.42) to jointly train for classification and bounding-box regression, using the approximate joint training approach described in Section 2.7.3.

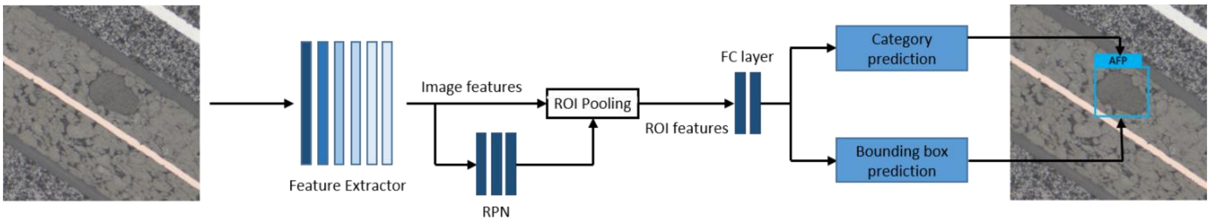


Figure 3-17: The Faster R-CNN model.

Single shot multibox detector based detection system

Compared with Faster R-CNN, SSD is conceptually simpler to understand and implement as it eliminates the region proposal generation steps and the subsequent resampling of feature step, thus making it quicker than Faster R-CNN at test time. As described in Section 2.7.4, SSD architecture uses feature maps from multiple layers of the feature extractor to detect objects of various sizes and aspect ratios. Similar to the Faster R-CNN, we compare the ResNet-50 architecture and the VGG-19 architecture for the base feature extractor network, removing the last two layers (global average pooling and fully connected layer) and appending four additional convolutional layers known as *Extra Feature Layers* with a decaying spatial resolution of depths [512, 512, 256, 256] respectively on top of the base network. Each multiscale feature block reduces the height and width of the feature map from the previous layer by half. So instead of using one feature map for predicting the classification scores and bounding box coordinates, features are extracted from these extra feature layers, in addition to feature maps from the last layer of the "conv4" and "conv5" blocks of the base feature extractor network to generate prediction maps of different resolution for multi-scale detection. In this way, large fine-grained feature maps at the earlier levels can detect small defects, while the smaller coarse-grained feature map closer to the output with large receptive fields can detect larger objects. The resolution of the detection is equal to the size of its feature map. For example, for an image of size 512 x 512, the network outputs six feature maps of resolutions 32 x 32 and 16 x 16 from the base feature extractor network and 8 x 8, 4 x 4, 2 x 2 and 1 x 1 from the extra feature layers respectively. For each location in the feature map, there are k default bounding boxes having different sizes and aspect ratios associated with each cell of the feature map. The default boxes are designed such that each feature map corresponds to a specific scale of default boxes with a predefined list of aspect ratios for each scale, as described in Section 2.7.4. k is set to [4, 6, 6, 6, 4, 4] for each of the six prediction layers, respectively, giving a total of $(32^2 \times 4 + 16^2 \times 6 + 8^2 \times 6 + 4^2 \times 6 + 2^2 \times 4 + 1^2 \times 4) = 6132$ anchor boxes were generated for each image at the six prediction layers. The minimum and maximum scales, s_{min} and s_{max} are set to 0.2 and 0.9 respectively. This means that the scale at the lowest prediction layer is 0.2, and the scale at the highest prediction layer is 0.9, while the other layers in between are regularly spaced. Then, for each of the bounding boxes, we need to compute c classification scores, including the background class and 4 offset values ($\Delta cx, \Delta cy, w$ and h) representing the offsets from the centre of the default box and its dimensions. Thus, each prediction is represented by $(c + 4)$ values. To achieve this, for a feature map with k default boxes per cell, we apply a prediction layer with 3 x 3 convolutional with $k * (c + 4)$ channels and padding of 1 to predict the categories and offsets of the default anchor boxes. By doing this, the height and width of the input and output of the convolutional layer remain the same.

However, since feature maps from multiple scales are used to generate anchor boxes to predict their categories and offsets, the prediction output at different scales could have different shapes because the number and shape of anchor boxes centred on the same element are different for feature maps of different scales. As a result, we need to first transform the output of the prediction layer into a consistent format and concatenate the predictions of multiple scales to simplify subsequent computation. Therefore, considering that the format of the prediction output is (batch size, number of channels, height, width), where the channel dimension contains the predictions for all anchor boxes having the same centre, we first transpose the channel dimension to the final dimension and convert the prediction results to the format (batch size, height x width x number of channels). Subsequently, the predictions are concatenated on the first dimension since all scales have the same batch size. In order to train the network, all the default boxes are matched to the ground truth boxes. First, each ground truth box is matched to the default box with the highest IoU overlap. Doing this ensures that each ground truth box is matched to at least one corresponding default box. Then each default box is matched to any ground truth box with an IoU overlap greater than 0.5. The default boxes which were not matched to any ground truth box are considered negative and contribute only to the confidence loss Equation (2.51), while the positively matched boxes contribute to both the confidence and localization loss. The final objective Equation (2.49) is given by combining these two losses, where the relative weighting of the confidence loss and localization loss α is obtained through cross validation. However, note that out of the 6132 default boxes, most of them are negative and including them all would lead to a severe class imbalance during training. Therefore, after matching the default boxes, they are sorted according to their confidence loss, and only the top ones are picked so that the ratio between the negatives and positives is at most 3:1, which helps stabilize training and leads to faster optimization.

3.10.4 Experiment settings for the defect localization networks

The dataset was randomly split into a training and testing set using an 80/20 split, as we did in the previous experiments. The training set contains 2,657 images with 3081 annotated defects, while the testing set consists of 639 images with 750 annotated defects. The distribution of the defect classes in the training and testing set is shown in Figure 3-19. Since all the images in the dataset have different sizes, during training of the Faster R-CNN model, the images are resized such that the smaller side has a dimension of 600 pixels and the larger side has a maximum size of 1000 pixels. For the SSD model, which requires all inputs to have a fixed size, all the images are resized to 512 x 512 pixels. The pixel values are rescaled in the range [0, 1]. It was difficult to train the object localization models on only this defect dataset in the initial experiments due to the relatively small size. Therefore, transfer learning was used to improve the model's accuracy and reduce the total training time taken for the model to converge. The main issue was that the localization model did not learn any useful features and simply overfitted the training dataset despite using various regularization techniques.

Nonetheless, there are various ways to apply transfer learning for object localization models. For instance, the feature extractor module of the localization model can be initialized with weights from feature extractor models that were trained on the ImageNet dataset or with the weights from our multiclass classification networks, while the rest of the network would be initialized with random weights. This way, the top layers of the localization network would already have a good starting point for extracting relevant features before training on our defect localization dataset. However, in order to prevent the large error signals in the early training iteration from effectively destroying the learned representation in the feature extractors since the rest of the network is initialized with random weights, all the weights of the feature extractor module are frozen, and just the weights of the localization module is fine-tuned. Afterwards, the feature extractor module weights are unfrozen then the whole network is jointly trained.

Alternatively, the weights of the entire object localization network can be initialized with pretrained weights from some other localization tasks such as the Pascal Visual Object Classes (VOC) [157] or Microsoft Common Object in Context (COCO) [158] dataset and fine-tune the weights of the entire model end-to-end. This ensures that the model is first initialized to localize common objects before training it to localize defects. Our initial experiments showed that initializing the object localization model with pretrained weights from a previous detection task tends to be more relevant and adaptive, as all the weights have been jointly learned. This makes fine-tuning the model converge significantly faster with higher accuracy than initializing only the feature extractor network with pretrained weights. Therefore, we adopt the second approach for all our subsequent experiments.

The hyperparameters used for training the Faster R-CNN model followed by those used for training the SSD model are described next. Following the same procedure as in the previous experiments, we apply five-fold cross-validation to evaluate the settings and tune the model's hyperparameters based on the result of the validation set. First, the training images are normalized by subtracting the mean and dividing by the standard deviation of the ImageNet dataset. The batch size for the Faster R-CNN model is set to 1, and the maximum number of epochs is set to 100. The network was trained using Nesterov accelerated gradient (NAG) descent as the optimizer, and the initial learning rate was set to 0.0005. The optimal learning rate was determined using a learning rate finder function which gradually increases the learning rate from a very small initial value until the training loss diverges. That is, training is done for one batch at a time, starting with a very small learning rate of 1×10^{-6} recording the training loss and gradually increasing it using a multiplier of 1.1 on every iteration until the training loss diverges. As shown in Figure 3-18, there is little change in the value of the loss for very small learning rates since the parameter updates are negligible. At a learning rate of 0.0001, the loss value starts to fall, and a drop in the loss value to about 0.001 can be seen, where the loss flattens and then starts to diverge. However, since the optimal initial learning rate should be as high as possible before the loss becomes unstable, the learning rate is chosen to be around the midpoint of these two values.

A second-degree polynomial scheduling function that gives a smooth decay gradually decreases the learning rate from the initial value of 0.0005 to 0 after 150000 iterations. The momentum and weight decay parameters are set to 0.9 and 0.0001, respectively. During training, the only data augmentation used was to flip the image horizontally with a probability of 0.5 randomly. For the SSD model, a batch size of 16 was used, and training was done for a maximum of 100 epochs. Here, the NAG optimizer was also used with the same weight-decay and momentum value but with a learning rate of 0.001, which was also discovered using the same learning rate finder function described earlier. Similarly, a second-degree polynomial schedule gradually decreased the learning rate from 0.001 to 0 after 20000 iterations. During the training of the SSD model, various data augmentations were applied, such as random colour jittering, randomly cropping areas around the defect location, randomly zooming with a probability of 0.5, randomly resizing the image, and randomly flipping the image horizontally with a probability of 0.5. These augmentations are only applied during the training phase, while the original images are used at test time. An early stopping value of 30 epochs was set, after which the model stops training if there is no improvement in the loss value. For the validation, the area of overlap (IOU) threshold between the predicted bounding box and ground truth for positive detection is set to 0.45, and the NMS threshold for overlapping bounding boxes is set to 0.5.

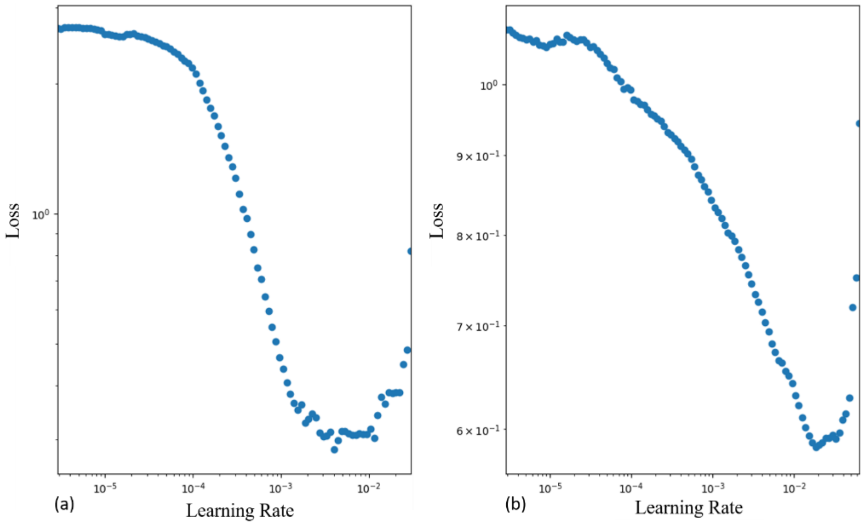


Figure 3-18: Graph showing how the loss value changes when increasing the value of the learning rate for (a) the Faster R-CNN network and (b) the SSD network.

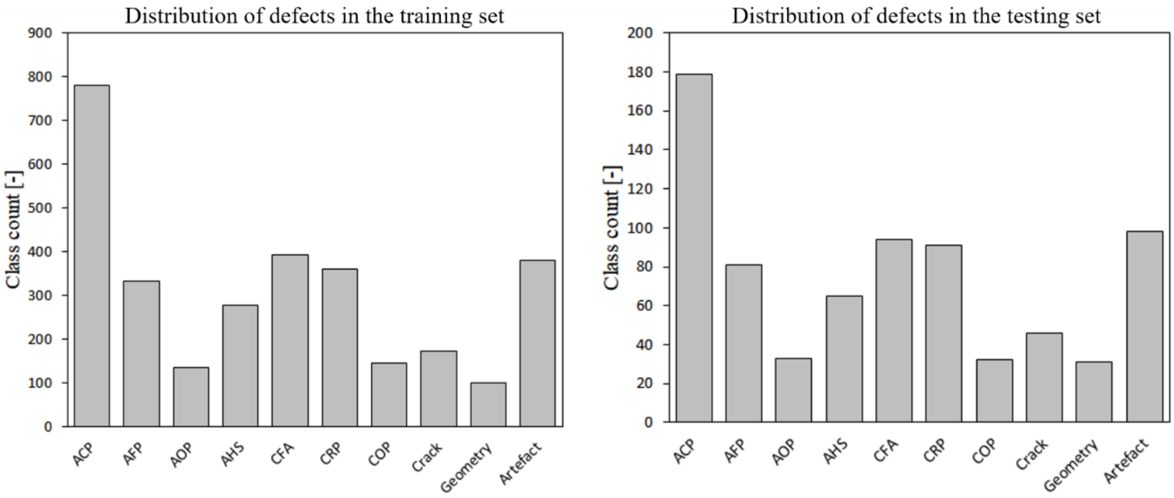


Figure 3-19: Distribution of defect classes in the training and testing set for the defect localization models.

3.11 Detecting defects in sintered NdFeB magnet

The following section develops a CNN-based generative pattern recognition algorithm based on variational autoencoders to detect defects in an unsupervised approach. Our objective is to develop a large-scale, high-dimensional anomaly detection algorithm that is sufficiently robust, such that it can generate an accurate model for data drawn from a wide range of probability distributions and is not overly affected by small departures from the trained model. Subsequently, based on the findings from the first approach, we improve the system's performance by using a conditional generative adversarial network (cGAN) to detect defects in the microstructure of sintered NdFeB magnet using only examples of normal structures without any prior information of defective cases.

3.11.1 Variational Autoencoder

As we described in Section 2.6.3, a VAE encodes an input image x to a latent vector $z = Encoder(x) \sim q(z|x)$ using an encoder network, and then decodes the latent vector z back to an image $\bar{x} \ = Decoder(z) \sim p(x|z)$ which is similar to the original image using the decoder network. Unlike a regular autoencoder, which maps the input onto a latent vector, a VAE maps the input data into the parameters of a probability distribution, such as the mean and variance of a Gaussian distribution. An important property of the VAE is the ability to control the distribution of the latent vector z learned by the network in the bottleneck layer, which is independent unit Gaussian random variables, i.e., $z \sim \mathcal{N}(0, 1)$. Thus, by minimizing the negative expected log-likelihood of each observation (pixel) in x using the reconstruction loss L_{rec} and the KL Divergence L_{kl} to regularize the encoder network to control the distribution of the latent variable $z,$ the network is able to learn a generative model of the underlying data distribution $p_{data}(x)$. Due to the constraint imposed on the latent vector z the network learns a compressed representation of the input data such that only samples obtained from the same underlying data distribution as the training set can be decoded back to the original input data. Thus, if a sample that does not belong to the same data generating distribution, such as an anomaly or a defect, is fed into the model, the latent vector cannot be meaningfully decoded back to the original input. Therefore, we can determine if an image contains a defect by comparing the reconstruction error between the input data and the decoded output. An illustration of the VAE is displayed in Figure 3-20.

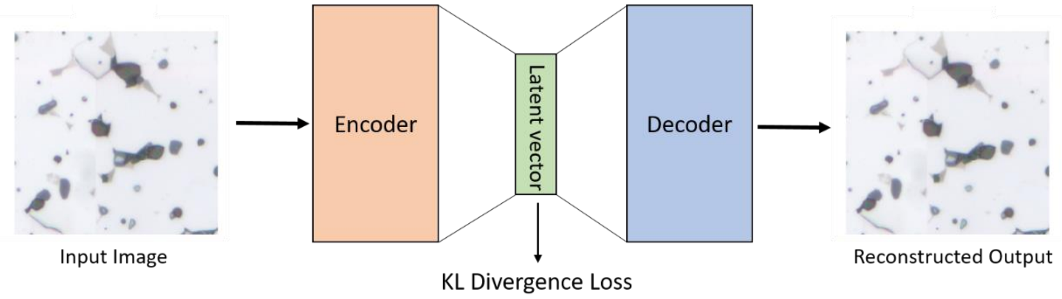


Figure 3-20: Illustration of a variational autoencoder network.

3.11.2 Dataset for training the variational autoencoder network

Since the VAE is trained in an unsupervised approach where the input to the model is also used as the target, obtaining a large amount of training data is straightforward since every single image does not need to be labelled. However, the training set must contain only images of normal samples. Therefore, the data set must be cleaned to ensure that it does not contain any images with defects since even a small amount of anomalies can significantly affect the model's performance. As illustrated in Figure 3-21, the training dataset was obtained by splitting two large cross-section images of magnet samples (HS17051_15 and HS17051_22) into 26,197 smaller patches of size 512 x 512 RGB colour images. Subsequently, any image containing defects is removed from the training dataset and set aside for testing. The testing set consists of 6000 images, of which only 50 samples contain defects or irregularities, while the other 5950 images are normal samples. Some examples of images with and without defects used for training and evaluating the model's performance are shown in Figure 3-22.

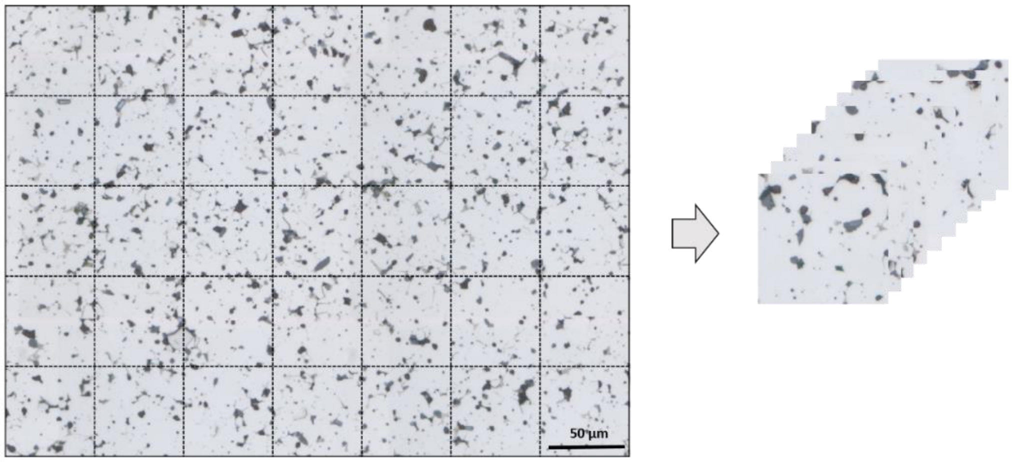
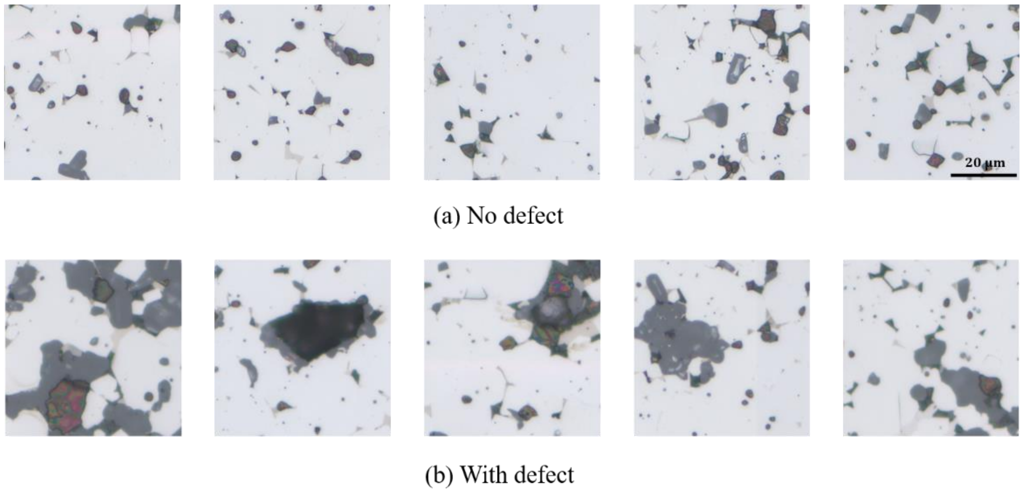


Figure 3-21: Shows a large image cross-section split into smaller patches of size 512 x 512 to obtain the training dataset.



(a) No defect

(b) With defect

Figure 3-22: (a) Shows normal examples of images from the training dataset, and (b) shows examples of images with defects included in the testing set.

3.11.3 Model architecture and implementation details

For the encoder network, which takes as input an image and outputs a set of parameters for specifying the conditional distribution of the latent variable z, we use five convolutional layers with 3 x 3 kernels and a stride of 2 to achieve spatial downsampling instead of using a deterministic spatial function such as the max pooling operation. Each convolutional layer is followed by a Leaky ReLU activation layer. Two fully connected output layers follow the last convolutional layer in the encoder network to parameterize the mean and variance used to compute the KL divergence loss and sample latent variable z. For the decoder network, which takes the latent sample z as input and outputs the parameters for a conditional distribution of the observation, the encoder architecture is mirrored by using a fully connected layer followed by five transpose convolutional layers for upsampling with 5 x 5 kernels using a stride of 2 with each layer followed by a Leaky ReLU activation. The output layer is a convolutional layer with 5 x 5 kernels using a stride of 1 with 3 output channels to map the input to the number of channels in the original image, followed by a Tanh activation function. No batch normalization layers were included to avoid destabilizing the network during training. The architecture of the variational autoencoder is shown in Figure 3-23. To train the VAE, the KL divergence loss L_{kl} and per pixel mean squared error L_{rec} were jointly minimized to obtain the total reconstruction loss, which is defined as:

$$L_{total} = \alpha L_{rec} + \beta L_{kl} \tag{3.11}$$

where α and β are weighting parameters for the image reconstruction and KL divergence, respectively.

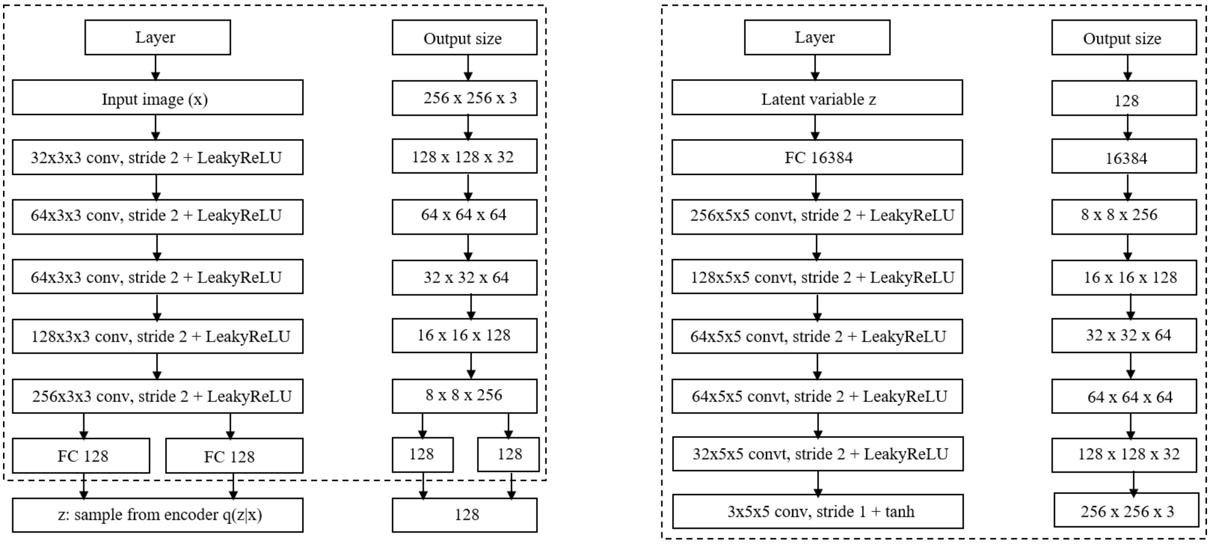

Layer
Output size
Input image (x)
256 x 256 x 3
32x3x3 conv, stride 2 + LeakyReLU
128 x 128 x 32
64x3x3 conv, stride 2 + LeakyReLU
64 x 64 x 64
64x3x3 conv, stride 2 + LeakyReLU
32 x 32 x 64
128x3x3 conv, stride 2 + LeakyReLU
16 x 16 x 128
256x3x3 conv, stride 2 + LeakyReLU
8 x 8 x 256
FC 128
FC 128
128
128
z: sample from encoder q(z|x)
128
Layer
Output size
Latent variable z
128
FC 16384
16384
256x5x5 convt, stride 2 + LeakyReLU
8 x 8 x 256
128x5x5 convt, stride 2 + LeakyReLU
16 x 16 x 128
64x5x5 convt, stride 2 + LeakyReLU
32 x 32 x 64
64x5x5 convt, stride 2 + LeakyReLU
64 x 64 x 64
32x5x5 convt, stride 2 + LeakyReLU
128 x 128 x 32
3x5x5 conv, stride 1 + tanh
256 x 256 x 3

Figure 3-23. The variational autoencoder network architecture. The encoder network is on the left side, and the decoder network is on the right.

3.11.4 Evaluation metrics for unsupervised defect detection models

In order to determine how well the network can reconstruct the input data, we need a way to meaningfully compare the similarity between the original input and the regenerated output.

Structural similarity index measure (SSIM) [159]. Unlike the mean squared error (MSE), which directly compares pixel-by-pixel differences and thus is not highly indicative of the perceived similarity between two images, SSIM measures the perceptual difference between two similar images based on the visible structures in the image while taking texture into account. This is often a better measure of a human perceptual judgement of image quality. The SSIM is calculated on various windows of an image, where the measure between two windows x and y of the same size $N \times N$ is:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.12}$$

where $\mu_x, \mu_y, \sigma_x^2, \sigma_y^2, \sigma_{xy}$ are the average, variance and covariance of x and y, respectively. $c_1 = (k_1L)^2, c_2 = (k_2L)^2$ are used to stabilize the division; L is the dynamic range of the pixel-values and $k_1 = 0.01$ and $k_2 = 0.03$. The resulting index is a decimal value between 0 and 1, where a value of 1 indicates that the two images are identical and 0 indicates no structural similarity.

3.11.5 Experiment settings for the VAE network

The input images are resized to 256 x 256 pixels during training, and the pixel values are rescaled between -1 and 1. The model is trained with a batch size of 16 for 100 epochs over the training dataset using RMSprop optimizer with a learning rate of 0.0001. The dimension of the latent vector z is set to 128. Other values such as 256 and 512 were also investigated but did not significantly improve performance. The loss weighting parameters α and β are set to 1 and 0.5, respectively. To generate a sample z for the decoder during training, we can sample from the latent distribution defined by the parameters output by the encoder for a given input observation x. However, sampling is a stochastic process, and therefore the gradient cannot be backpropagated through a random node. Therefore, a commonly used reparameterization trick is applied to approximate z using the encoder parameters and an auxiliary independent random variable $\epsilon \sim \mathcal{N}(0, 1)$ as follows $z = \mu + \sigma \odot \epsilon$ where μ and σ represent the mean and standard deviation of a Gaussian distribution respectively and $\odot$ is the element-wise product. Thus we make the model trainable by learning the mean and variance of the distribution in the encoder network through backpropagation while maintaining stochasticity through ϵ. The experimental results are presented in Section 4.3.1.

3.12 Conditional Generative Adversarial Network

After analysing the results of the VAE model, we observed that the model could detect most images with defects correctly; however, the model also produced many false positives.

Therefore, since each magnet sample we analyse consists of several thousand tile images, it is essential to minimize the false-positive rate for the system to be helpful in practice. Thus, after examining the false positive results from the VAE model, the results showed that image tiles with more objects or patterns tend to produce a higher reconstruction error even though they do not contain any defects. Nevertheless, this is expected since the input image is highly compressed into a 128-dimensional latent vector space, resulting in some spatial and structural information loss. On the other hand, if a non-generative model such as a simple autoencoder were used, the model would be equally able to reconstruct images with defects and without defects. Hence, we need an approach that can preserve the spatial information of structures in the image and a loss function that produces sharp and realistic images to reduce the number of false positives.

GANs can generate realistic images by learning a function that tries to classify if an image is real or fake (synthesized by the generator) while simultaneously training a generative model to minimize this function. As a result, unrealistic or blurry images that are obviously fake are not acceptable as the discriminator model easily detects them. However, since GANs learn a generative model of the training data in order to produce new samples from the same underlying data distribution, which are different from the input data, we need to learn a conditional generative model of the input data to ensure that the generated outputs bear some close resemblance to the input. Conditional GANs (cGANs) learn a conditional generative model of the input data making them particularly suitable for many image-to-image translation tasks, where we condition an input image and generate a corresponding output image. In contrast to the per-pixel mean squared error loss function used in the previous approach, where each output pixel is considered independently from all other pixels for a given input image, conditional GANs learn a *structured loss* which penalizes the joint formation of the output image as a whole [160]. As described in Section 2.6.4, GANs learn a mapping from random noise vector z to output image $y, G : z \rightarrow y$ while cGANs learn a mapping from observed image x and random noise vector z, to output $y, G : \{x, z\} \rightarrow y$. To achieve this, the generator G is trained to generate realistic outputs that cannot be easily differentiated from real images by an adversarially trained discriminator, D, which is simultaneously trained to detect real from fake images produced by the generator. This procedure is illustrated in Figure 3-24. Therefore, the objective of the conditional GAN can be expressed as follows:

$$L_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))], \tag{3.13}$$

where G tries to minimize this objective while D tries to maximize it, that is:

$$G^* = \arg \min_{\mathrm{G}} \max_{D} L_{cGAN}(G, D). \tag{3.14}$$

In addition, a $L1$ loss is included in the training objective to capture low-frequency details, which encourages less blurring compared to $L2$ loss. This also forces the generator to produce outputs that are near the ground truth label:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]. \tag{3.15}$$

Thus our final objective is given as follows:

$$G^{*} = \arg \min_{G} \max_{D} L_{c}GAN(G, D) + \lambda\mathcal{L}_{L1}(G). \qquad (3.16)$$

where λ is a weighting term hyperparameter to determine the contribution of the L_1 loss.

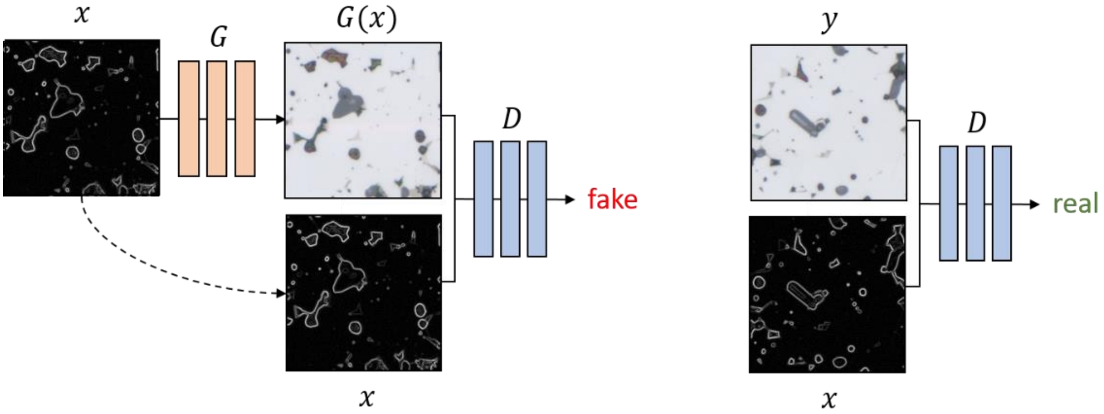
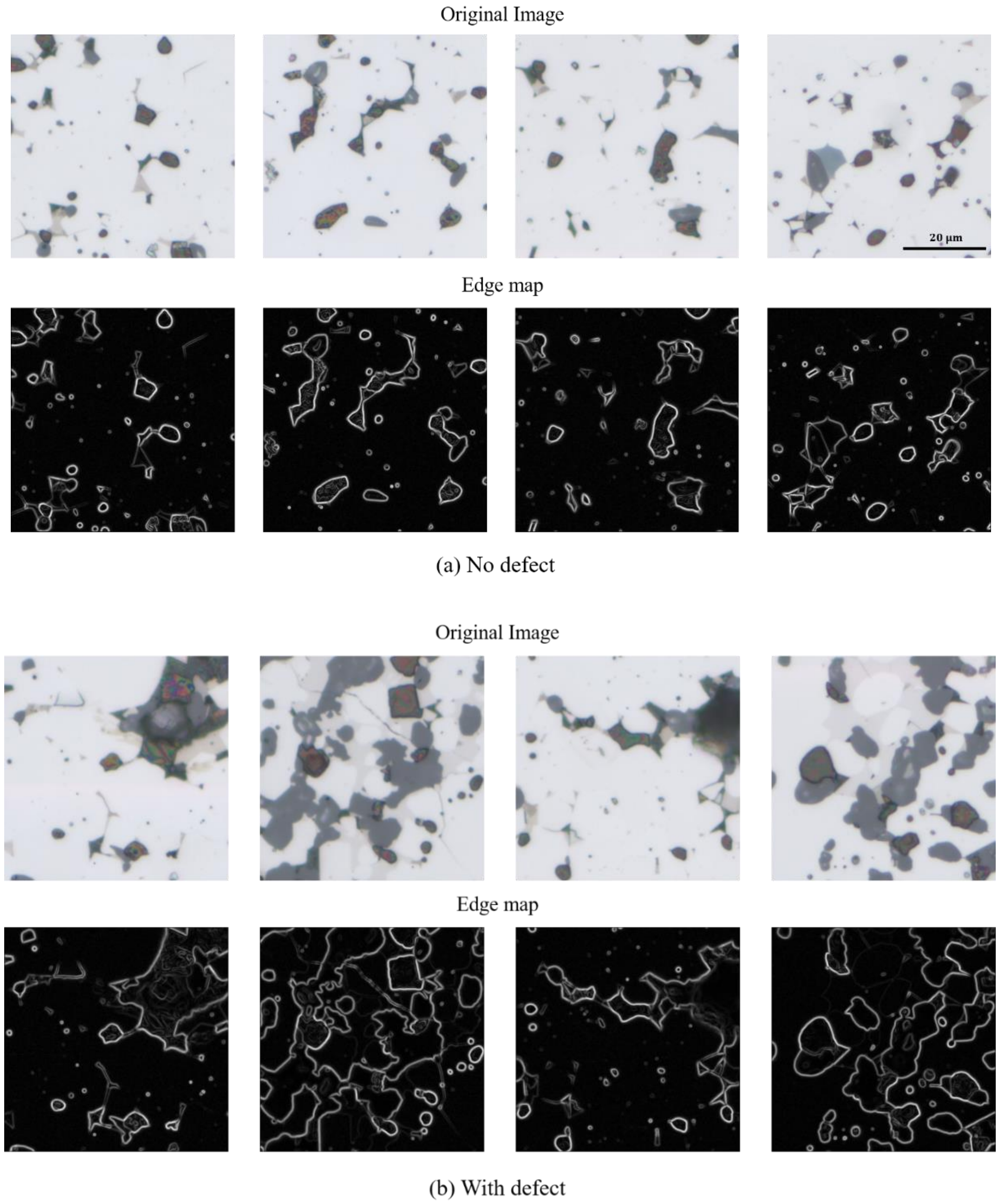


Figure 3-24: Illustrates the procedure for training a conditional GAN to map edges to images. The discriminator, D, learns to classify between the real and fake (produced by the generator) edge-image pairs, while the generator, G, learns to deceive the discriminator by generating convincing samples for given edge maps.

In essence, the aim here is to train a conditional generative model to map images from one domain to another with the overall goal of detecting images with defects by comparing the reconstruction errors. Therefore, a unique representation of the original image is needed to condition the model to deterministically generate an accurate reconstruction of normal images as the desired output. A simple albeit not obvious idea is to use the edges of the structure in the image as the conditional input and train the model to map edges to images, as illustrated in Figure 3-24. The intuition behind this is that the morphology of the edges of the structures in the image will provide subtle cues to the model of what type of structure is most likely to generate such an outline. Moreover, since the model never sees the complete picture, it cannot simply copy the input to the output and is forced to learn a good mapping. As a result, the model can simply focus on learning the relationship between edges and structures and does not have to encode the positions of each structure in the image, which is a more difficult problem. Furthermore, since the outline of regions with defects would, in principle, be anomalous compared with the other structures in the image, the model will have no representation for mapping such edges to meaning structures while accurately mapping all the normal ones. Therefore, this will likely result in a higher reconstruction error solely for regions with defects, and less error will be due to compression artefacts.

3.12.1 Dataset for training the cGAN network

For training the conditional GAN, we used the same dataset used to train the VAE model in the previous section with one fundamental difference. Since our aim in this section is to train a model to map edges to images, these edge maps are generated by applying a Sobel filter to the original images. This was done in three steps: first, we convert the images to greyscale, then apply the Sobel filter to the images and finally normalize the edge maps to values between 0 and 1. Figure 3-25 shows some images with their corresponding edge maps for examples with and without defects. Note that no images with defects were used for training the model, and they are only included for evaluation purposes.



(a) No defect

(b) With defect

Figure 3-25: (a) Shows some examples of normal images and the corresponding edge maps. (b) Shows some examples with defects and the corresponding edge maps.

3.12.2 Model architecture and implementation details

This problem can be described as an image-to-image translation problem where the input and output simply differ in surface appearance, but both renderings are essentially representations of the same underlying structures. For instance, there is a large amount of low-level information shared between the input edge map and the output image, such as the shape, texture and position of certain structures, and it would be highly desirable to pass this information directly across the network. Therefore, we base the architecture for the generator and discriminator on those in [160].

Generator architecture

As such, the architecture for the generator network is based on these considerations. Many previous approaches [161]–[164] to similar problems have used an encoder-decoder network like the one described in Section 2.4.2. We use a similar design here, but the generator is given the means to circumvent the bottleneck layer by adding skip connections following the general shape of a U-Net architecture [165]. In particular, skip connections are added between each layer i and layer $n-i,$ where n is the total number of layer. The skip connections concatenate activation outputs of all channels at layer i with those at layer $n-i$ to shuffle low-level features closer to the input layer across the network. The details of the U-Net encoder-decoder architecture is shown in Figure 3-26. In all the convolutional layers kernels of size $4 \times 4,$ stride 2 and padding 1 are used followed by a Leaky ReLU activation with a slope of 0.2 in the encoder network and a ReLU activation in the decoder network. After the last layer in the decoder network, a convolution layer with 3 output channels is applied to map the output to the number of channels in the original image followed by a Tanh activation function. Convolutions in the encoder network downsample the input by a factor of 2, while in the decoder they upsample by a factor of 2 using a convolution transpose operation. Except for the first convolution layer in the encoder, batch normalization (BN) with momentum 0.1 is applied after all the other convolution layers. Noise is provided to the generator network by applying dropout with a rate of 50% to several layers of the decoder.

Discriminator architecture

The architecture of the discriminator is much simpler than that of the generator since it essentially just needs to classify if an image is real or fake (synthesised by the generator). Moreover, considering that a L_1 loss is included in the generator objective to capture low-frequency details, the discriminator only needs to model high-frequency structures. Accordingly, to capture high frequencies, it is sufficient to limit the attention of the discriminator to only penalize structures in local image patches [160]. That is, the discriminator only needs to classify each $N \times N$ patch in an image as either real or generated and we average all the responses to calculate the final output. There are several advantages to classifying patches compared to a single binary output. For one, the training dynamics of the model are more stable and, therefore, faster to train due to the rich feedback from the patch discriminator compared to regular GANs, which can be very difficult to train. In addition, a fixed-size path discriminator can be applied to arbitrarily large images. Analogous to the generator model, kernel of size $4 \times 4,$ stride 2 and padding 1 are used in all the convolution layers, followed by

Leaky ReLU activation. Except for the first layer, batch normalization is applied after all the other layers. Lastly, a 1 × 1 convolution is used to map the channels in the last layer to a 1-dimensional output followed by a sigmoid activation function. The details of the discriminator model architecture are shown in Figure 3-27.

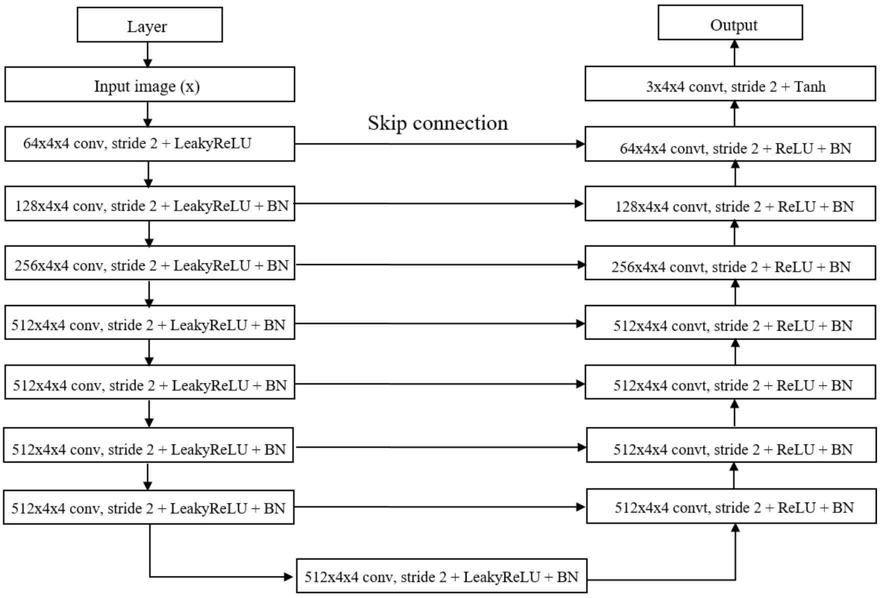

Layer
Input image (x)
64x4x4 conv, stride 2 + LeakyReLU
128x4x4 conv, stride 2 + LeakyReLU + BN
256x4x4 conv, stride 2 + LeakyReLU + BN
512x4x4 conv, stride 2 + LeakyReLU + BN
512x4x4 conv, stride 2 + LeakyReLU + BN
512x4x4 conv, stride 2 + LeakyReLU + BN
512x4x4 conv, stride 2 + LeakyReLU + BN
512x4x4 conv, stride 2 + LeakyReLU + BN
Skip connection
Output
3x4x4 convt, stride 2 + Tanh
64x4x4 convt, stride 2 + ReLU + BN
128x4x4 convt, stride 2 + ReLU + BN
256x4x4 convt, stride 2 + ReLU + BN
512x4x4 convt, stride 2 + ReLU + BN
512x4x4 convt, stride 2 + ReLU + BN
512x4x4 convt, stride 2 + ReLU + BN
512x4x4 convt, stride 2 + ReLU + BN

Figure 3-26: U-Net generator network architecture. “conv” stands for the convolution operation, “convt” stands for the convolution transpose operation, and “BN” stands for bach normalization.

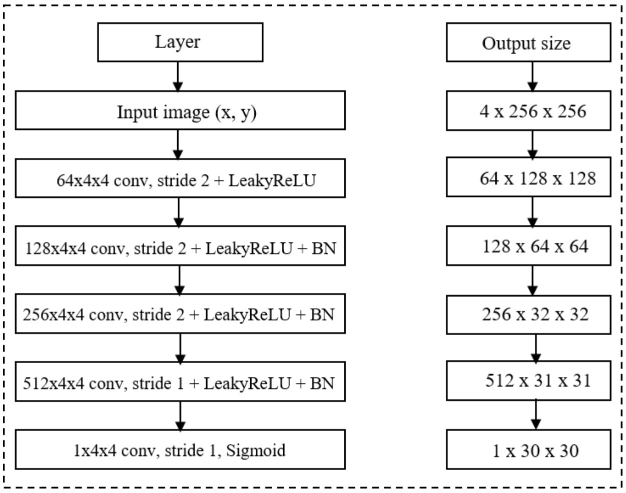

Layer
Input image (x, y)
64x4x4 conv, stride 2 + LeakyReLU
128x4x4 conv, stride 2 + LeakyReLU + BN
256x4x4 conv, stride 2 + LeakyReLU + BN
512x4x4 conv, stride 1 + LeakyReLU + BN
1x4x4 conv, stride 1, Sigmoid
Output size
4 x 256 x 256
64 x 128 x 128
128 x 64 x 64
256 x 32 x 32
512 x 31 x 31
1 x 30 x 30

Figure 3-27: Discriminator network architecture. “conv” and “BN” stand for the convolution and batch normalization operation, respectively.

3.12.3 Experiment settings for the cGAN network

Similar to the training of the VAE model, all the images are resized to 256×256 pixels. The pixel values for the ground truth images are normalized to values between -1 and 1, while those of the edge map have values between 0 and 1. For training the networks, the standard approach from [133] was followed, where we alternate between one gradient descent step for D, then one step for G. First, a batch of fake images is fed to D and the loss is calculated using binary cross-entropy loss function, then a batch of real images is fed to D, and the average of the two losses is taken. As suggested in the original GAN paper, G is trained to maximize $\log D\big(x, G(x,z)\big)$ rather than trained to minimize $\log\Big(1 - D\big(x, G(x,z)\big)\Big)$. After some initial investigations on the hyperparameter settings, the model was trained with a batch size of 64 for 100 epochs using Adam optimizer with a learning rate of 0.00002 and momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$. Various values for λ were investigated but a value of 100 produced the best results. Both networks were trained from scratch with the weights initialized from a Gaussian distribution with mean 0 and standard deviation of 0.02. No data augmentations were applied during training as this only destabilises the networks. The experimental results are presented in Section 4.3.2.

Chapter 4

Results

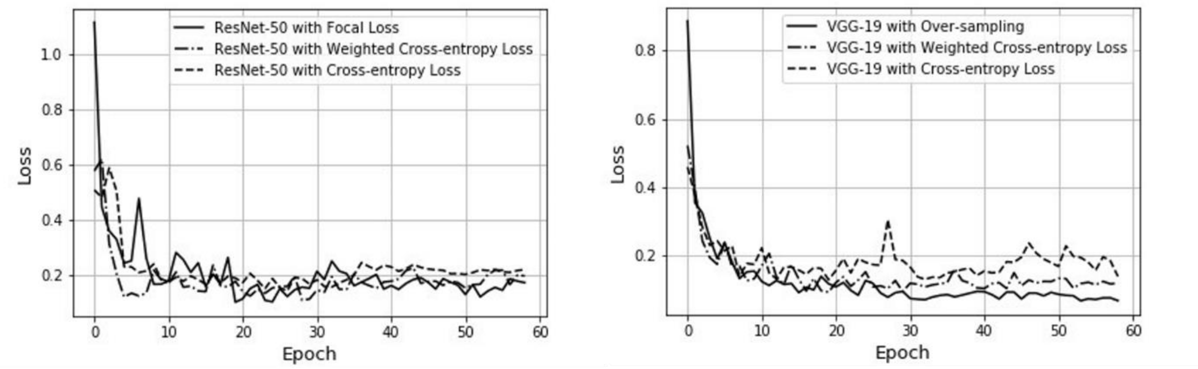
4.1 Experimental evaluation of supervised defect detection models for Li-ion batteries

We conducted several experiments while developing our defect detection model for Li-ion batteries, and this section presents the results of those experiments. First, the results of the binary classifier model developed in Section 3.8 are presented, followed by the multiclass classifier model developed in Section 3.9. Subsequently, visualizations of the model's activations are presented using a class activation mapping technique, which helps identify regions in the image where the model focuses its attention when making a prediction. Lastly, experimental results of the defect localisation models developed in Section 3.10 are presented.

4.1.1 Experimental results of binary classifier models

First, let us look at the effect of the focal loss function hyperparameters for various values of α and γ on the performance of the two models. Although the bests value of γ reported in [120] was 2, values 1 and 5 were also investigated. The optimal value of γ for the ResNet-50 model was found to be 2 and for the VGG-19 model 1. Values of α in the range of 1 to 4 with a step size of 1 were also compared. The F_1-score, accuracy, and AUROC are shown in Table 4-1. The F_1-score for both models increases as the value of α increases, with ResNet-50 achieving a peak value at $\alpha = 3$ and VGG-19 at $\alpha = 4$. ResNet-50 with focal loss function surpassed the model with standard cross-entropy loss function in three out of four cases, while VGG-19 with focal loss function did so in two out of 4 cases. Both ResNet-50 and VGG-19 models trained with standard cross-entropy loss function achieved the same performance with an F_1-score of 0.94.

In contrast, ResNet-50 trained with the weighted cross-entropy loss function achieved an F_1-score of 0.976, outperforming the VGG-19 model F_1-score 0.961 trained with the same loss function. The performance of the two models using both re-sampling techniques and standard cross-entropy loss function were also investigated (see Table 4-1). Overall, the ResNet-50 model with focal loss function achieved the best performance with an F_1-score of 0.976, slightly outperforming the same model with weighted cross-entropy, as it could detect more positive samples. Nevertheless, it can be seen that employing the cost-sensitive learning methods improved the model performances of both models compared with simply using the standard cross-entropy loss function. In both cases, the over-sampling technique performed better than the under-sampling technique, probably due to fewer training samples left after undersampling the majority class. In fact, for the VGG-19 models, using standard cross-entropy in combination with the over-sampling technique achieved a better performance than both the weighted cross-entropy loss and the focal loss function with an F_1-score of 0.968 and only slightly behind the best ResNet model. However, this is at a higher computational cost since the over-sampling technique increases the number of samples in the training phase. The validation losses for both models are shown in Figure 4-1.



(a) Comparison of the validation losses produced by the top-2 ResNet-50 models with model using cross-entropy loss function

(b) Comparison of the validation losses produced by the top-2 VGG-19 models with model using cross-entropy loss function

Figure 4-1: Shows the validation loss of the top-performing models compared with the model trained using the standard cross-entropy loss function.

Table 4-1: Shows the F_1-score, Accuracy, AUROC and AUPRC achieved by the binary classification models for each training technique and each hyperparameter in the focal loss. The best performances are shown in bold text.

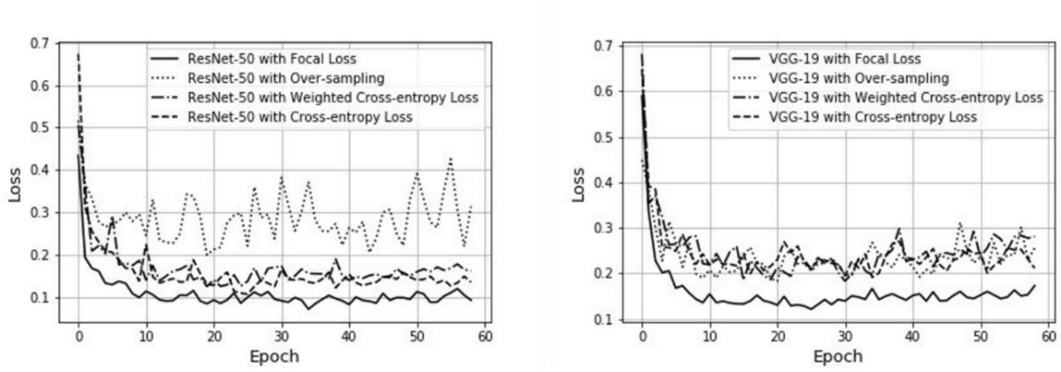
Model architecture	Method	Hyper-parameter		F_1-score	Accuracy	AUROC	AUPRC	TP	FN	FP	TN
		α	γ		(%)						
ResNet-50	Focal loss	1	2	0.852	94.1	0.952	0.901	57	7	13	231
		2		0.962	98.2	0.992	0.984	59	5	**0**	**244**
		3		**0.976**	**99.0**	0.997	0.987	**62**	**2**	1	243
		4		0.954	97.7	0.989	0.973	61	3	3	241
	Cross entropy	-	-	0.944	97.7	0.986	0.966	59	5	2	242
	Weighted CE	-	-	**0.976**	**99.0**	**0.998**	**0.993**	61	3	**0**	**244**
	Under-sampling	-	-	0.918	96.8	0.971	0.949	56	8	2	242
	Over-sampling	-	-	0.952	98.1	0.990	0.982	60	4	2	242
VGG-19	Focal loss	1	1	0.918	96.8	0.987	0.971	56	8	2	242
		2		0.938	96.9	0.981	0.973	59	5	3	241
		3		0.944	97.7	0.983	0.970	59	5	2	242
		4		0.960	98.4	0.978	0.972	60	4	1	243
	Cross entropy	-	-	0.944	97.7	0.984	0.969	59	5	2	242
	Weighted CE	-	-	0.961	98.4	0.983	0.976	**62**	**2**	3	241
	Under-sampling	-	-	0.838	92.9	0.970	0.934	57	7	15	229
	Over-sampling	-	-	0.968	98.7	0.975	0.971	61	3	1	243

4.1.2 Experimental results of multiclass classifier models

Similar to the previous experiments, the effect of various values of α and γ hyperparameters on the focal loss function was investigated. For both the ResNet-50 and VGG-19 models, the optimal value for γ was found to be 1. Again, the value of α was varied from 1 to 4 in step sizes of 1. However, the results did not show similar improvements in the model performance as the value of α was increased. Thus, the optimal value for the ResNet-50 model was found to be $\alpha = 1$ and for the VGG-19 model $\alpha = 2$. The F_1-score, accuracy and AUROC are shown in Table 4-2, and the loss curves are shown in Figure 4-2. ResNet-50 with focal loss function surpassed the model with standard cross-entropy loss function in 2 out of 4 cases, while VGG-19 with focal loss surpassed in 1 out of 4 cases. Surprisingly, both the ResNet-50 model trained with a weighted cross-entropy loss function and the oversampling technique achieved an F_1-score of 0.972, which is slightly below the model trained with a standard cross-entropy loss function F_1-score 0.973. Overall, ResNet-50 with focal loss function with $\alpha = 1$ and $\gamma = 1$ achieved the best performance of 0.976 F_1-score, outperforming the model trained with standard cross-entropy loss. The graphs of the ROC and PR curves of the top-performing model are shown in Figure 4-3 and Figure 4-4, respectively. Lastly, the confusion matrix of the ResNet-50 models and the VGG models developed using the various training techniques are presented in Figure 4-5 and Figure 4-6, respectively. These results are discussed in Section 5.1.1.

Table 4-2: Shows the F_1-score, Accuracy, AUROC and AUPRC achieved by the multiclass classification models for each training technique and each hyperparameter in the focal loss. The best performances are shown in bold text.

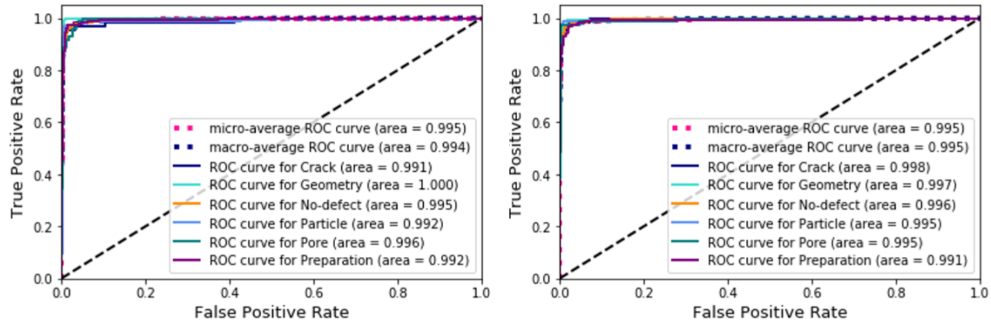
Model architecture	Method	Hyper-parameter		F_1-score	Accuracy	AUROC	AUPRC
		α	γ		(%)		
ResNet-50	Focal loss	1	1	**0.976**	**97.6**	**0.995**	0.985
		2		0.974	97.4	**0.995**	0.981
		3		0.965	96.5	0.992	0.977
		4		0.963	96.2	0.991	0.966
	Cross entropy	-	-	0.973	97.3	0.994	0.980
	Weighted CE	-	-	0.972	97.2	0.994	**0.986**
	Over-sampling	-	-	0.972	97.2	**0.995**	0.985
VGG-19	Focal loss	1	1	0.955	95.6	0.990	0.977
		2		0.964	96.4	0.992	0.980
		3		0.957	95.7	0.993	0.983
		4		0.949	95.0	0.990	0.972
	Cross entropy	-	-	0.958	95.8	0.992	0.979
	Weighted CE	-	-	0.959	96.0	0.991	0.978
	Over-sampling	-	-	0.961	96.1	0.990	0.980



(a) Comparison of the validation losses produced by the ResNet-50 models from the various training techniques

(b) Comparison of the validation losses produced by the VGG-19 models from the various training techniques

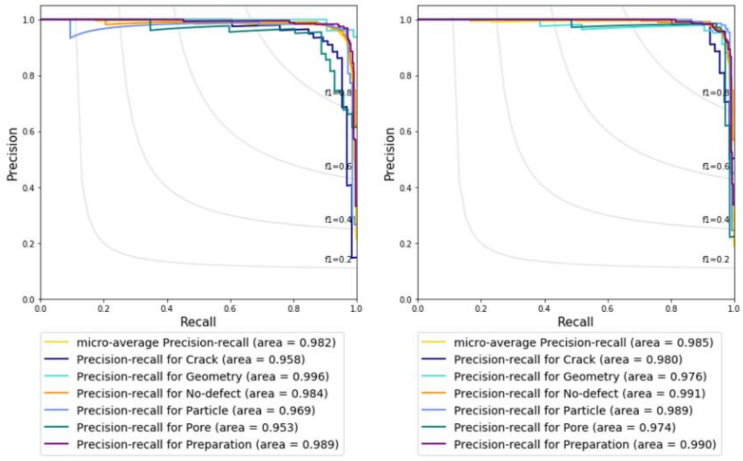
Figure 4-2: Shows the validation loss of the top-performing models compared with the model trained using the standard cross-entropy loss function.



(a) ResNet-50 with cross-entropy loss function

(b) ResNet-50 with focal loss function ($\alpha = 1.0 \; \gamma = 1.0$)

Figure 4-3: Comparison of ROC curves for each defect category achieved by the model with cross-entropy loss function (a) and the model with focal loss function (b). The model achieved a similar performance in terms of AUROC with both loss functions.

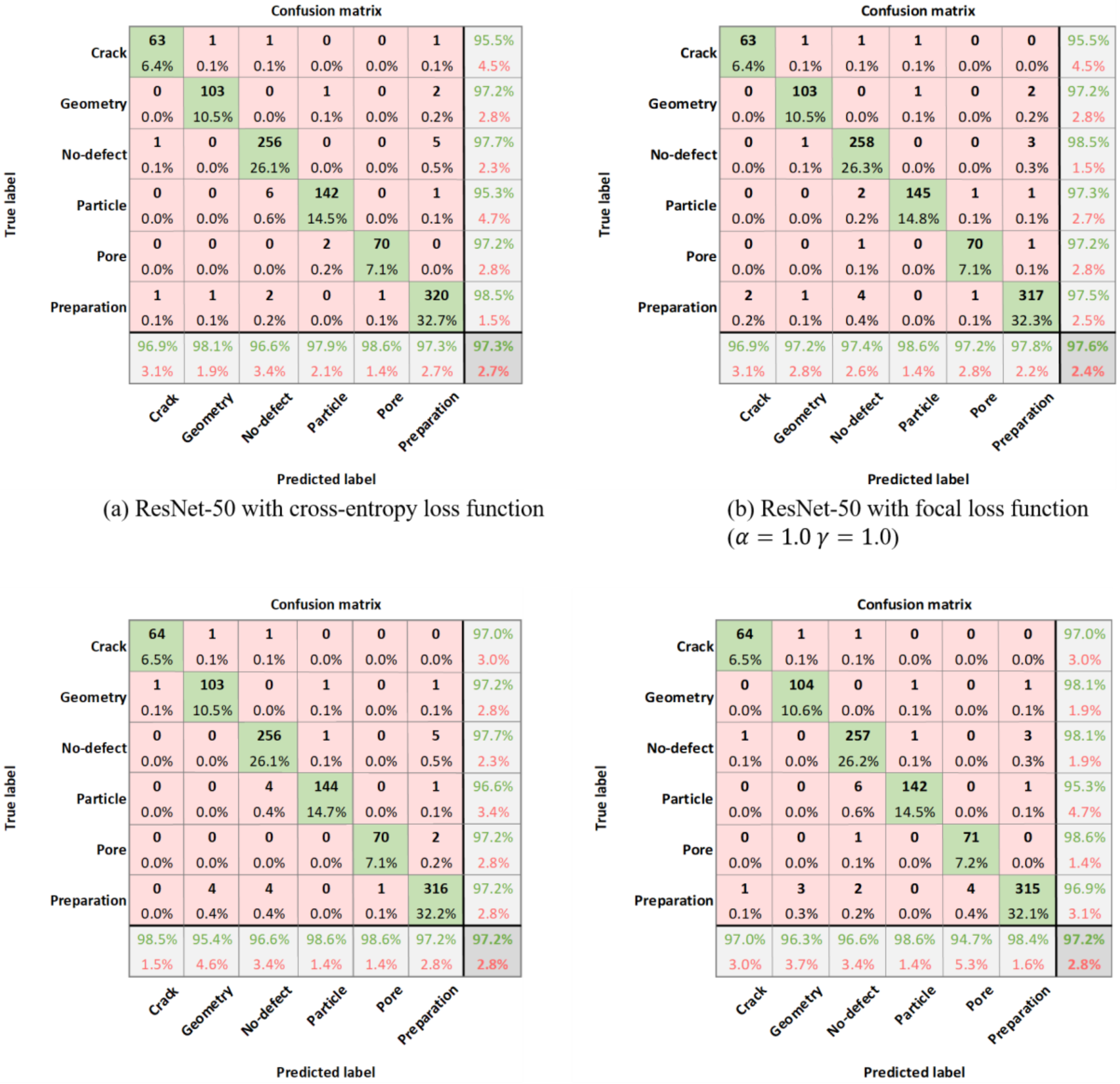


(a) ResNet-50 with cross-entropy loss function

(b) ResNet-50 with focal loss function ($\alpha = 1.0$ $\gamma = 1.0$)

Figure 4-4: Comparison of PR curves for each defect category achieved by the model with cross-entropy loss function (a) and the model with focal loss function (b). The model trained

with focal loss achieved a higher AUPRC for all classes except the geometric irregularity class than those achieved with a standard cross-entropy loss function.


True label
Predicted label

Confusion matrix

	Crack	Geometry	No-defect	Particle	Pore	Preparation	
Crack	63 6.4%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	1 0.1%	95.5% 4.5%
Geometry	0 0.0%	103 10.5%	0 0.0%	1 0.1%	0 0.0%	2 0.2%	97.2% 2.8%
No-defect	1 0.1%	0 0.0%	256 26.1%	0 0.0%	0 0.0%	5 0.5%	97.7% 2.3%
Particle	0 0.0%	0 0.0%	6 0.6%	142 14.5%	0 0.0%	1 0.1%	95.3% 4.7%
Pore	0 0.0%	0 0.0%	0 0.0%	2 0.2%	70 7.1%	0 0.0%	97.2% 2.8%
Preparation	1 0.1%	1 0.1%	2 0.2%	0 0.0%	1 0.1%	320 32.7%	98.5% 1.5%
	96.9% 3.1%	98.1% 1.9%	96.6% 3.4%	97.9% 2.1%	98.6% 1.4%	97.3% 2.7%	97.3% 2.7%

(a) ResNet-50 with cross-entropy loss function

Confusion matrix

	Crack	Geometry	No-defect	Particle	Pore	Preparation	
Crack	63 6.4%	1 0.1%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	95.5% 4.5%
Geometry	0 0.0%	103 10.5%	0 0.0%	1 0.1%	0 0.0%	2 0.2%	97.2% 2.8%
No-defect	0 0.0%	1 0.1%	258 26.3%	0 0.0%	0 0.0%	3 0.3%	98.5% 1.5%
Particle	0 0.0%	0 0.0%	2 0.2%	145 14.8%	1 0.1%	1 0.1%	97.3% 2.7%
Pore	0 0.0%	0 0.0%	1 0.1%	0 0.0%	70 7.1%	1 0.1%	97.2% 2.8%
Preparation	2 0.2%	1 0.1%	4 0.4%	0 0.0%	1 0.1%	317 32.3%	97.5% 2.5%
	96.9% 3.1%	97.2% 2.8%	97.4% 2.6%	98.6% 1.4%	97.2% 2.8%	97.8% 2.2%	97.6% 2.4%

(b) ResNet-50 with focal loss function ($\alpha = 1.0\ \gamma = 1.0$)

Confusion matrix

	Crack	Geometry	No-defect	Particle	Pore	Preparation	
Crack	64 6.5%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	97.0% 3.0%
Geometry	1 0.1%	103 10.5%	0 0.0%	1 0.1%	0 0.0%	1 0.1%	97.2% 2.8%
No-defect	0 0.0%	0 0.0%	256 26.1%	1 0.1%	0 0.0%	5 0.5%	97.7% 2.3%
Particle	0 0.0%	0 0.0%	4 0.4%	144 14.7%	0 0.0%	1 0.1%	96.6% 3.4%
Pore	0 0.0%	0 0.0%	0 0.0%	0 0.0%	70 7.1%	2 0.2%	97.2% 2.8%
Preparation	0 0.0%	4 0.4%	4 0.4%	0 0.0%	1 0.1%	316 32.2%	97.2% 2.8%
	98.5% 1.5%	95.4% 4.6%	96.6% 3.4%	98.6% 1.4%	98.6% 1.4%	97.2% 2.8%	97.2% 2.8%

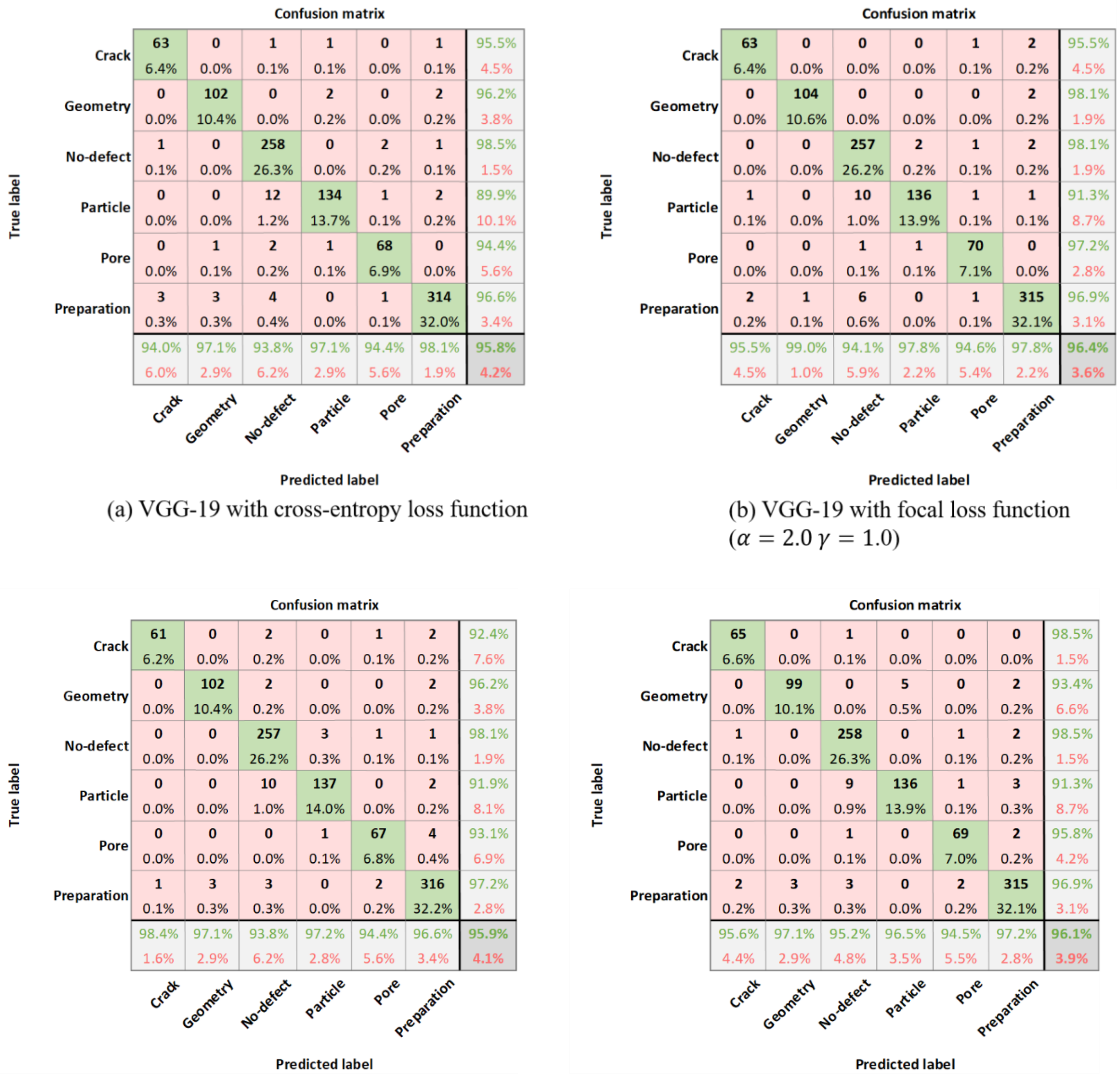
(c) ResNet-50 with weighted cross-entropy loss function

Confusion matrix

	Crack	Geometry	No-defect	Particle	Pore	Preparation	
Crack	64 6.5%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	97.0% 3.0%
Geometry	0 0.0%	104 10.6%	0 0.0%	1 0.1%	0 0.0%	1 0.1%	98.1% 1.9%
No-defect	1 0.1%	0 0.0%	257 26.2%	1 0.1%	0 0.0%	3 0.3%	98.1% 1.9%
Particle	0 0.0%	0 0.0%	6 0.6%	142 14.5%	0 0.0%	1 0.1%	95.3% 4.7%
Pore	0 0.0%	0 0.0%	1 0.1%	0 0.0%	71 7.2%	0 0.0%	98.6% 1.4%
Preparation	1 0.1%	3 0.3%	2 0.2%	0 0.0%	4 0.4%	315 32.1%	96.9% 3.1%
	97.0% 3.0%	96.3% 3.7%	96.6% 3.4%	98.6% 1.4%	94.7% 5.3%	98.4% 1.6%	97.2% 2.8%

(d) ResNet-50 with over-sampling technique

Figure 4-5: Comparison of the confusion matrices for the ResNet-50 models using the various training techniques.



Confusion matrix
True label
Predicted label

	Crack	Geometry	No-defect	Particle	Pore	Preparation	
Crack	63 6.4%	0 0.0%	1 0.1%	1 0.1%	0 0.0%	1 0.1%	95.5% 4.5%
Geometry	0 0.0%	102 10.4%	0 0.0%	2 0.2%	0 0.0%	2 0.2%	96.2% 3.8%
No-defect	1 0.1%	0 0.0%	258 26.3%	0 0.0%	2 0.2%	1 0.1%	98.5% 1.5%
Particle	0 0.0%	0 0.0%	12 1.2%	134 13.7%	1 0.1%	2 0.2%	89.9% 10.1%
Pore	0 0.0%	1 0.1%	2 0.2%	1 0.1%	68 6.9%	0 0.0%	94.4% 5.6%
Preparation	3 0.3%	3 0.3%	4 0.4%	0 0.0%	1 0.1%	314 32.0%	96.6% 3.4%
	94.0% 6.0%	97.1% 2.9%	93.8% 6.2%	97.1% 2.9%	94.4% 5.6%	98.1% 1.9%	95.8% 4.2%

(a) VGG-19 with cross-entropy loss function

	Crack	Geometry	No-defect	Particle	Pore	Preparation	
Crack	63 6.4%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	2 0.2%	95.5% 4.5%
Geometry	0 0.0%	104 10.6%	0 0.0%	0 0.0%	0 0.0%	2 0.2%	98.1% 1.9%
No-defect	0 0.0%	0 0.0%	257 26.2%	2 0.2%	1 0.1%	2 0.2%	98.1% 1.9%
Particle	1 0.1%	0 0.0%	10 1.0%	136 13.9%	1 0.1%	1 0.1%	91.3% 8.7%
Pore	0 0.0%	0 0.0%	1 0.1%	1 0.1%	70 7.1%	0 0.0%	97.2% 2.8%
Preparation	2 0.2%	1 0.1%	6 0.6%	0 0.0%	1 0.1%	315 32.1%	96.9% 3.1%
	95.5% 4.5%	99.0% 1.0%	94.1% 5.9%	97.8% 2.2%	94.6% 5.4%	97.8% 2.2%	96.4% 3.6%

(b) VGG-19 with focal loss function ($\alpha = 2.0\ \gamma = 1.0$)

	Crack	Geometry	No-defect	Particle	Pore	Preparation	
Crack	61 6.2%	0 0.0%	2 0.2%	0 0.0%	1 0.1%	2 0.2%	92.4% 7.6%
Geometry	0 0.0%	102 10.4%	2 0.2%	0 0.0%	0 0.0%	2 0.2%	96.2% 3.8%
No-defect	0 0.0%	0 0.0%	257 26.2%	3 0.3%	1 0.1%	1 0.1%	98.1% 1.9%
Particle	0 0.0%	0 0.0%	10 1.0%	137 14.0%	0 0.0%	2 0.2%	91.9% 8.1%
Pore	0 0.0%	0 0.0%	0 0.0%	1 0.1%	67 6.8%	4 0.4%	93.1% 6.9%
Preparation	1 0.1%	3 0.3%	3 0.3%	0 0.0%	2 0.2%	316 32.2%	97.2% 2.8%
	98.4% 1.6%	97.1% 2.9%	93.8% 6.2%	97.2% 2.8%	94.4% 5.6%	96.6% 3.4%	95.9% 4.1%

(c) VGG-19 with weighted cross-entropy loss function

	Crack	Geometry	No-defect	Particle	Pore	Preparation	
Crack	65 6.6%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	98.5% 1.5%
Geometry	0 0.0%	99 10.1%	0 0.0%	5 0.5%	0 0.0%	2 0.2%	93.4% 6.6%
No-defect	1 0.1%	0 0.0%	258 26.3%	0 0.0%	1 0.1%	2 0.2%	98.5% 1.5%
Particle	0 0.0%	0 0.0%	9 0.9%	136 13.9%	1 0.1%	3 0.3%	91.3% 8.7%
Pore	0 0.0%	0 0.0%	1 0.1%	0 0.0%	69 7.0%	2 0.2%	95.8% 4.2%
Preparation	2 0.2%	3 0.3%	3 0.3%	0 0.0%	2 0.2%	315 32.1%	96.9% 3.1%
	95.6% 4.4%	97.1% 2.9%	95.2% 4.8%	96.5% 3.5%	94.5% 5.5%	97.2% 2.8%	96.1% 3.9%

(d) VGG-19 with over-sampling technique

Figure 4-6: Comparison of the confusion matrices for the VGG-19 models using the various training techniques.

4.1.3 Visualization of the class activation mapping

Up to this point, only the structures of the defect detection models, the number of defect classes and the evaluation functions of the models have been defined. It would be insightful to know what features the model learns from the data or given an image which parts lead the model to arrive at a particular decision. This would allow us to understand better the remaining sources of error in the model's decision process and focus our attention on those cases.

This section will show that despite training the network on only image-level labels, the global average pooling (GAP) layer [166] enables the convolutional network to accurately localize the discriminative image regions through a technique called *class activation mapping* (CAM)

[167]. A class activation map of a specific image category indicates the discriminative regions the CNN can identify. Although the global average pooling was originally proposed as a structural regularizer to prevent overfitting during training, it encodes a generic localizable deep representation that reveals the implicit attention of the CNN on an image. In other words, the global average pooling layer enforces the learning of category-level feature maps in the network and encourages the network to identify the complete extent of the object. Naturally, we expect that the largest activations will be observed in the feature map, which corresponds to the ground-truth label of the input image, which is directly enforced by global average pooling. Furthermore, it is also expected that the strongest activations appear approximately at the same location as the object in the original image within the feature map of the ground truth category. Therefore, through visualization of these feature maps, it is possible to understand where in the image the network focuses its attention when making a prediction by highlighting exactly which regions of the image are important for discrimination.

Also, with this approach, a simple form of object localization can be done by directly adapting these feature maps without training on any bounding box annotations. The procedure for generating the CAM is illustrated in Figure 4-7. An input image is fed to the network, and on the convolutional feature maps just before the final output layer, we perform global average pooling and use those as features for the fully connected layer to make the final class prediction. The global average pooling layer outputs the spatial average of the feature map for each unit at the last convolutional layer. Thus, to obtain the class activation maps, simply compute a weighted sum of the feature maps of the last convolutional layer and upsample the result to the size of the input image. Overlaying the resulting heatmap on the original image makes it possible to identify the most relevant regions for that particular category. Next, the CAM visualization for each defect class from the best ResNet-50 model trained using the focal loss function is shown. Afterwards, the visualizations for some misclassified examples are shown, and lastly, we show how this visualization technique can be used for object localization.

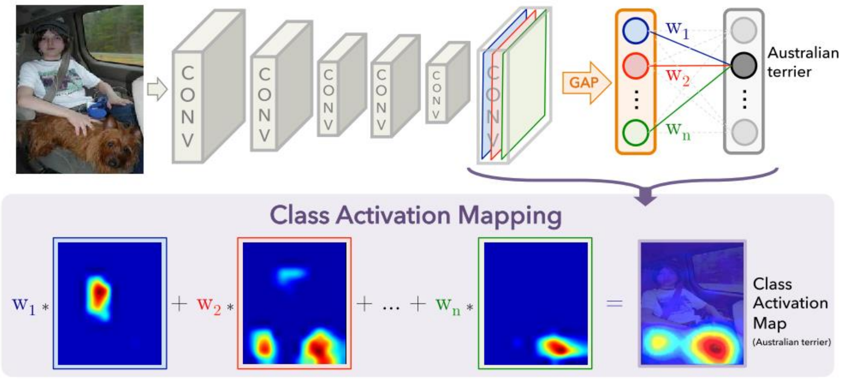


Figure 4-7: Illustrates the Class Activation Mapping where the predicted class score in the output layer is projected back to the previous convolutional layer to generate the CAM to highlight the class-specific discriminative regions [167].

Using the above approach, we show in Figure 4-8 some examples of the CAM outputs for each category of defects. The results show that only the discriminative regions used for classification are highlighted (in red) with remarkable accuracy. For example, the first image with two cracked regions in the anode material shows that the network focuses its attention on both cracks. Likewise, for the example of geometric irregularity, the first image shows that the network focuses its attention on the gap between the two electrodes, while in the second image, it focuses on the absence of the other half of the cathode coating material. Moreover, looking at the examples with no defects, the majority of the image is highlighted, which shows that for an image to be categorized as having no defects, the network essentially has to focus its attention on the entire image as one would expect, as opposed to focusing on specific parts of the image like in the other examples. In general, by training the network on just categorical information, the network can discover the discriminative region in the image and accurately localize the extent of these regions without being trained explicitly on any bounding box annotation. These visualizations demonstrate the effectiveness of the proposed approach for accurately detecting defects in complex microstructural images of Li-ion batteries.

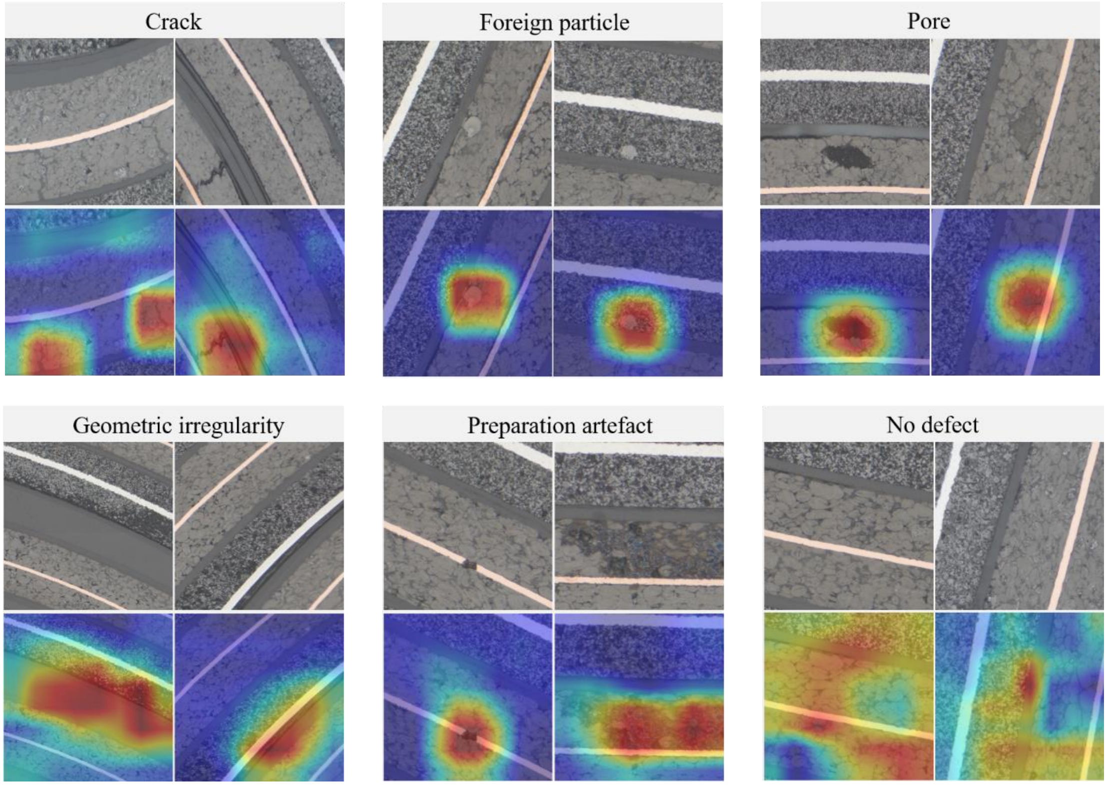


Figure 4-8: The CAMs for each defect class. The heat maps highlight the discriminative image regions used for image classification. Bright red colour indicates the strongest activations for a particular class, and blue regions indicate weak activation. Notice how the strongest activations are centred on the defects while the entire image is highlighted for the images with no defect.

Figure 4-9 shows examples of images that were misclassified by the best model and the top-3 predicted probabilities. By examining the result of the model predictions, we can observe two things: firstly, there are at least two or more types of defects in the same image for most misclassified examples. For example, in the first image, there is a crack in the top right corner of the cathode active material, there are some breakouts of active material in the cathode due to the sample preparation process, and the second half of the cathode active material is missing, which can be categorized as a geometric irregularity. In fact, for the first image in Figure 4-9, the model predicted a probability of 0.483 for geometric irregularity, 0.480 for preparation artefact and 0.024 for crack as the top three choices. However, a patch can only be assigned a single class label; therefore, the model is forced to pick the class with the highest predicted probability. Nonetheless, the results show that the ground truth label is consistently among the top-3 predicted probabilities, although due to the hard labels used for training (1 for true labels and 0 for everything else), the model is more likely to assign a high probability to one class and a low probability to all the other classes. This observation is one of the main reasons for developing an object localization model to detect multiple defects within the same image.

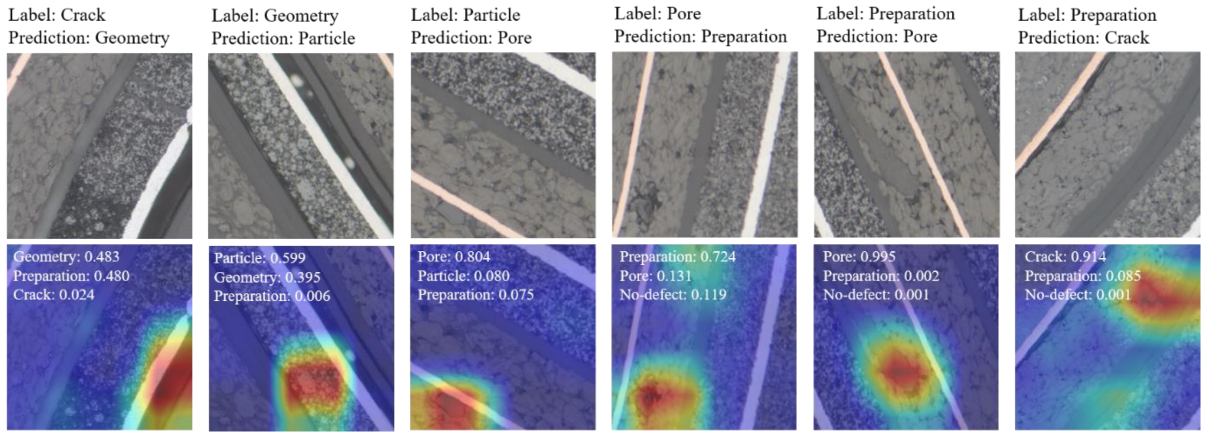


Figure 4-9: Examples of the class activation maps of some misclassified images. The predicted class and the score of the top-3 predictions are shown above each CAM image. Bright red colour indicates the strongest activation for a particular class, and blue indicates weak activation.

In order to perform object localization, tight-fitting bounding boxes need to be placed around the object of interest as well as the associated object category. A simple approach can be achieved through the class activation maps. Using the CAMs, a bounding box can be generated by applying a simple thresholding technique to segment the heatmap. To implement this, first, segment the regions with a value above 45% of the maximum value of the CAM, then generate a bounding box surrounding the largest connected component in the segmentation map. Figure 4-10 shows some example bounding boxes generated using this technique. The result is quite impressive, considering the model was only trained on categorical labels without any bounding box annotations, which can be time-consuming to acquire. For example, in the first image in Figure 4-10, three distinct cracks in the active material coatings can be detected by segmenting the regions with the highest activation level. However, for smaller objects, the size of the generated bounding boxes tends to be overestimated compared to the ground truth annotations, while those of larger objects are more closely fitting. Nonetheless, this technique provides a simple approach for improving the estimation of defect distribution and a method for estimating the size of the defects.

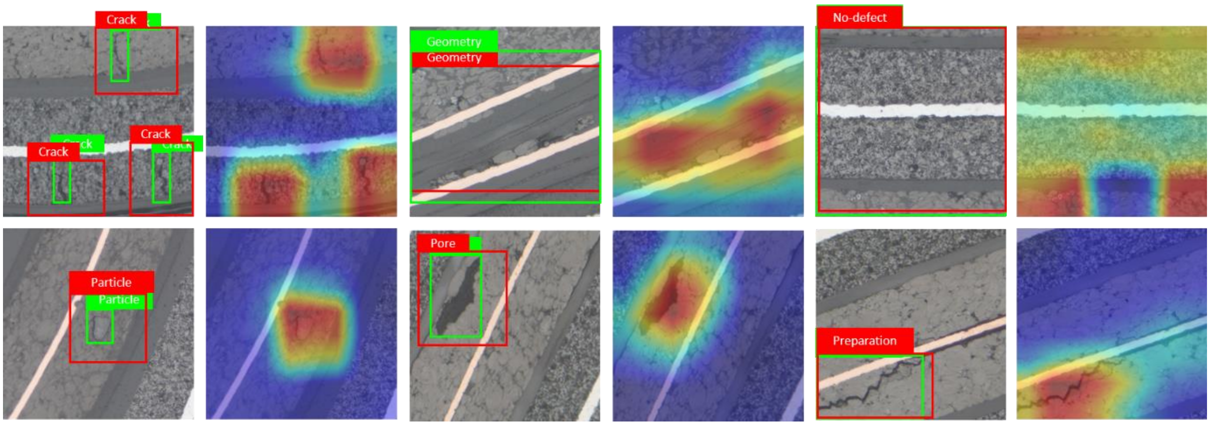


Figure 4-10: Examples of defect localization using the heatmap generated from the CAMs. The ground truth boxes are green, and the predicted bounding boxes from the class activation map are in red.

4.1.4 Experimental results of defect localization models

This section compares the performance of each model architecture using the mean of average precision (mAP) score on the testing set shown in Table 4-3. The evaluation speed of each architecture when using either the CPU or the GPU is also presented. For this, each image in the test data set is processed individually, and we report the average evaluation time per image, including the time taken to pre-process and post-process each image. Some example outputs from the trained model are shown in Figure 4-11. The models predict tight-fitting bounding boxes with high confidence around the detected defects irrespective of the size of the defect or the number of defects in the same image. Overall, the Faster R-CNN model with the ResNet-50 feature extractor achieved the best performance with an mAP of 0.87 on the testing set. While the Faster R-CNN model with VGG-19 feature extractor achieved an mAP of 0.83, and the SSD model with ResNet-50 and VGG-19 feature extractors achieved an mAP of 0.81 and 0.78, respectively.

In general, the defect localization models using the ResNet feature extractor performed much better than those using the VGG feature extractors, which is not surprising as the ResNet architecture was shown in our previous experiments to outperform the VGG architecture on the defect classification tasks. However, the evaluation time varies significantly between the networks, as shown in Table 4-3. The SSD model with ResNet-50 feature extractor is the fastest, with an evaluation time of 1.07 seconds per image using the CPU and 0.12 seconds per image on the GPU. The Faster R-CNN model with VGG-19 feature extractor is the slowest, requiring 23.46 seconds per image on the CPU and 0.52 seconds per image on the GPU. This is not surprising as the SSD architecture was originally designed to prioritize speed over classification accuracy. Nevertheless, the result is interesting because the evaluation times seem to be inversely correlated with the mAP, thus showing the speed/accuracy tradeoff described in [168]. The average precision (AP) for each of the defect categories of each network architecture is shown in Table 4-4. The results show that the lowest average precision values were obtained on the geometry and artefact classes, respectively. This can be attributed primarily to the fact that compared with the other defect classes, both the geometry and artefact classes tend to occupy a larger proportion of an image when present, and lack a clearly defined boundary, thus making it difficult for the network to predict a bounding box which closely matches the ground truth label. Furthermore, when compared with the other defect classes, there are many variations in the data set as to what is categorized as an artefact or geometric irregularity, which would obviously impact the performance of the models. Some examples of such predictions are shown in Figure 4-15. Here the Faster R-CNN model with ResNet-50 feature extractor predicted the class label correctly; however, the coordinates of the bounding boxes are incorrectly predicted, resulting in a low IoU score.

On the other hand, we observe high AP values for classes with homogeneous structures and well-defined boundaries, such as the AHS class and the foreign particle inclusions. Figure 4-16 provides three examples where the network makes false detections. In the first two examples, the network makes two predictions on the same defect region, a true positive and a false positive detection. The network correctly predicts the defect as an artefact with a high score; however, the score for the second prediction is also higher than the detection threshold. In the third example, the network predicts a score that is lower than the detection threshold; hence the defect was not detected in this case. The training loss for the fine-tuning of the network weights is shown in Figure 4-12. The results show that the ResNet variant achieved a lower loss value for both defect localisation networks and converged faster, which is particularly obvious when comparing the SSD network with the ResNet-50 feature extractor to the one with the VGG-19 feature extractor. Also, note that the SSD networks take a much smaller number of training steps to converge than the Faster R-CNN models. This is mainly because the Faster R-CNN network uses a batch size of 1, and in a lot of the images in our dataset, there is a very high ratio of background to foreground class, causing the network to learn the desired features much slower due to the imbalance. In contrast, the SSD networks can train with a larger batch size (a batch size of 16 was used during training) which reduces the imbalance in the data set and thus leads to faster convergence.

Next, the influence of the training data size on the model performance is evaluated. Many tasks involving deep learning usually take a large amount of labelled training data to achieve a good performance, either for image classification or image segmentation. Therefore, to compare the data efficiencies and requirements of the various model architectures, we train the defect localization models several times, each with different amounts of training data and observe how this affects each model's performance. Figure 4-13 shows how the training data size affects the mAP of each defect localisation model given the same testing set. The result shows that the defect detection accuracy increases linearly with the size of the data set, and extrapolating the result suggests that with a larger training data set, a higher mAP could be achieved. For the SSD network, the mAP increases significantly when the training data size reaches 2500 images.

Thus, it appears that the SSD networks require more training data to achieve comparable performance to the Faster R-CNN networks.

Earlier, we observed that there appears to be a tradeoff between the speed and accuracy of the object detection models. In particular, the number of region proposals generated by the region proposal network is known to affect the speed and accuracy of the Faster R-CNN architecture, whereby increasing the number of region proposals generated increases the chance that all defects will be found; however, this also increases the computational requirements of the network. In order to determine the optimal amount of region proposals required during test time without a significant drop in the detection accuracy of the model, several experiments were conducted by varying the number of proposals generated by the RPN from 50 to a maximum of 6000 and observing the effect on the model performance. Figure 4-14 shows the relationship between mAP, evaluation time, and the number of region proposals. Based on these results, the optimal number of region proposals that provides a good balance between speed and precision is 3000. In other words, we can reduce the number of region proposals generated by the RPN by half, which effectively reduces the evaluation time from 0.438 seconds to 0.341 seconds per image while maintaining the same mAP.

Lastly, two additional training methods for leveraging transfer learning are compared to investigate the influence of transfer learning on the training dynamics of the models and how it can help achieve faster convergence. The defect detection networks are trained with randomly assigned weights in the feature extraction layers using the Xavier initialization in the first training method. In the second approach, the feature extraction layers of the networks are initialized using weights pretrained on the ImageNet dataset. While in the third approach, the entire network is initialized with weights pretrained on the Pascal VOC dataset as described in Section 3.10.4. The result of each training approach is shown in Table 4-5, where each trained model is evaluated on the same testing set. The training method that does not leverage transfer learning achieved the lowest mAP and took the highest number of epochs to converge of all the models. In the second training approach, where the feature extractors are initialized with weights pretrained on ImageNet, the models achieve higher performance while taking fewer epochs to converge compared with the first training approach. Finally, the third training scheme that takes full advantage of transfer learning achieves the best performance of the three training methods while taking the least number of epochs to converge. These experiments clearly demonstrate how transfer learning helps the model generalize better to unseen data, especially when the size of the training set is relatively small.

Table 4-3: Shows the mean average precision and evaluation time per image on the testing set for each model on the defect detection task.

Architecture	Feature extractor	Evaluation time per image using CPU [s]	Evaluation time per image using GPU [s]	mAP
Faster R-CNN	ResNet-50	21.571	0.438	**0.87**
Faster R-CNN	VGG-19	23.462	0.517	0.83
SSD	ResNet-50	**1.073**	**0.121**	0.81
SSD	VGG-19	2.672	0.133	0.78

Table 4-4: Compares the average precision scores of each model for the ten defect categories.

Architecture	Feature extractor	Average precision									
		ACP	AFP	AOP	AHS	CFA	CRP	COP	Crack	Geometry	Artefact
Faster R-CNN	ResNet-50	**0.964**	**0.909**	0.851	**0.947**	**0.922**	**0.881**	**0.963**	0.801	0.715	**0.764**
Faster R-CNN	VGG-19	0.923	0.841	**0.855**	0.915	0.869	0.841	0.914	0.775	0.704	0.711
SSD	ResNet-50	0.862	0.832	0.825	0.837	0.833	0.832	0.813	**0.816**	**0.724**	0.720
SSD	VGG-19	0.824	0.813	0.793	0.811	0.791	0.717	0.802	0.783	0.719	0.704

Table 4-5: Shows the effect of different initialization schemes on the performance of the defect localization models. It demonstrates the benefits of transfer learning on model generalization and enables faster convergence.

		Mean Average Precision (mAP)		
Architecture	Feature extractor	Xavier Initialization (Random)	ImageNet Pretrained Weights	Pascal VOC Pretrained Weights
Faster R-CNN	ResNet-50	0.82, epochs 51	0.84, epochs 49	0.87, epochs 36
Faster R-CNN	VGG-19	0.80, epochs 55	0.81, epochs 52	0.83, epochs 36
SSD	ResNet-50	0.77, epochs 96	0.80, epochs 80	0.81, epochs 68
SSD	VGG-19	0.76, epochs 99	0.76, epochs 96	0.78, epochs 85

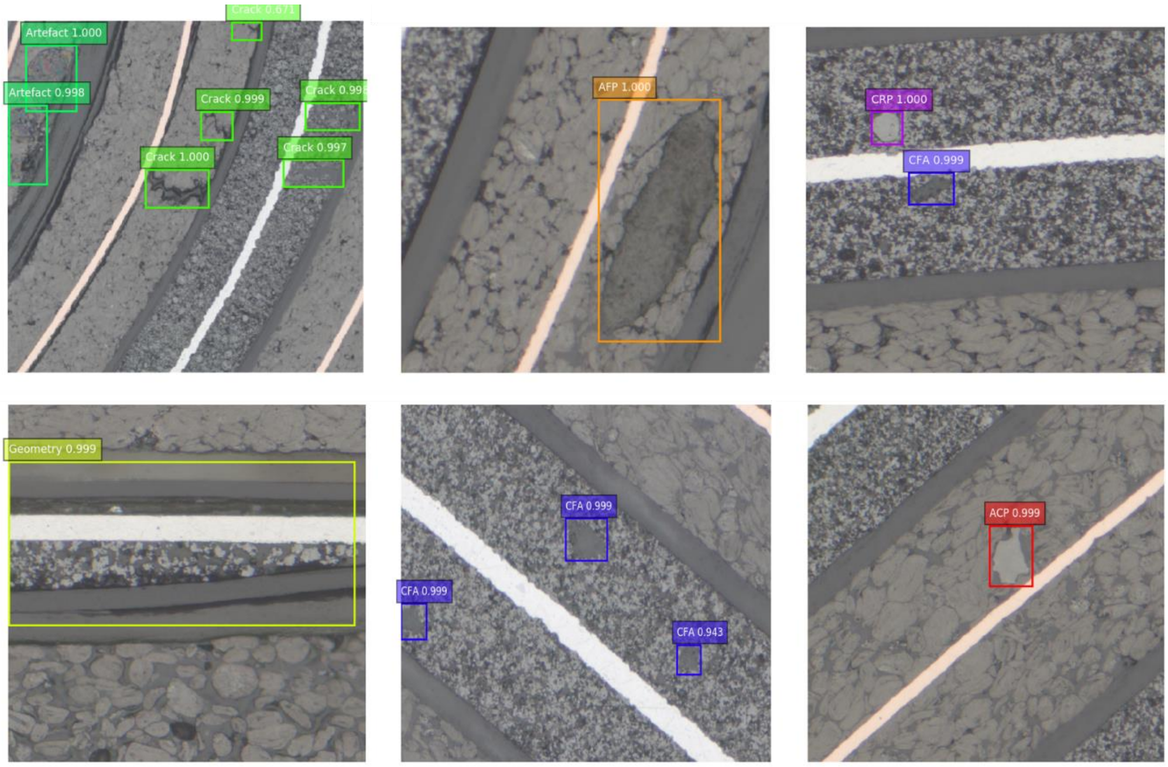
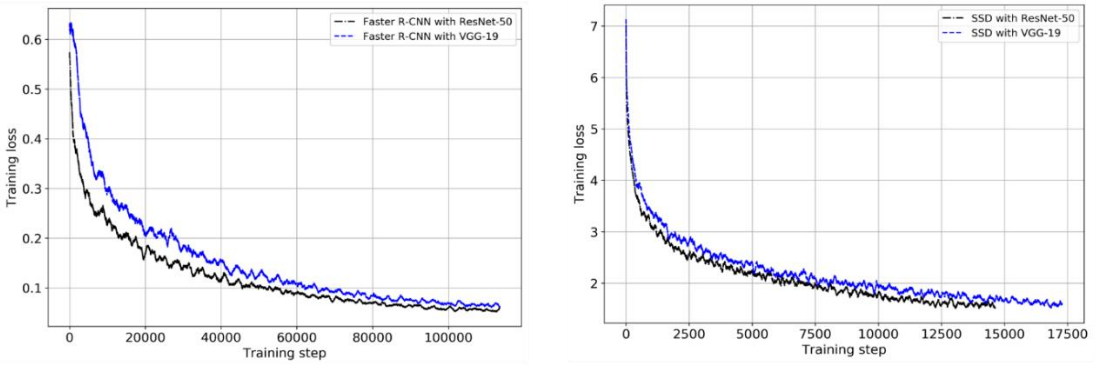


Figure 4-11: Examples of true positive defect detections from the trained Faster R-CNN model with ResNet-50 feature extractor.



(a) Training loss (smoothed) for the Faster R-CNN defect localization networks during the fine-tuning process.

(b) Training loss (smoothed) for the SSD defect localization networks during the fine-tuning process.

Figure 4-12: Smoothed training loss for the defect localization networks during the fine-tuning process. Note that the loss function for the Faster R-CNN networks is different from that of the SSD networks; thus, the relative magnitude of the loss is not relevant.

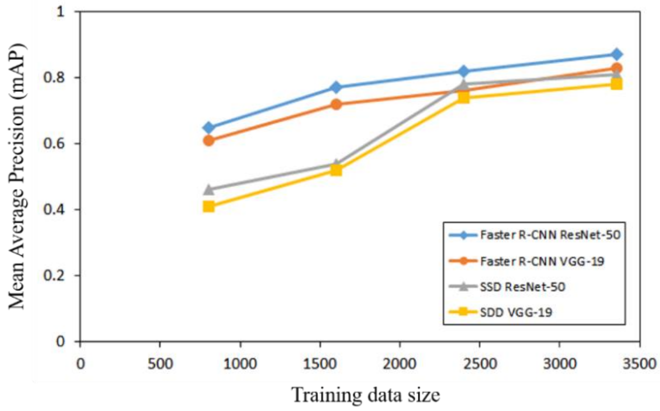


Figure 4-13: Shows the mean average precision (mAP) on the same testing set for different training data sizes for each defect localization network.

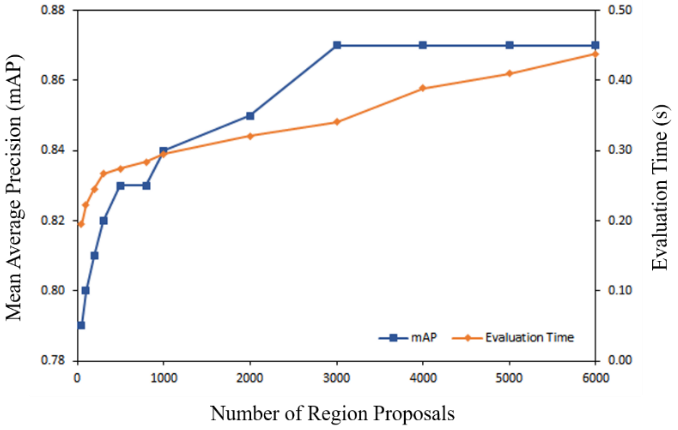


Figure 4-14: Relationship between the mean average precision, evaluation time and the number of region proposals for the Faster R-CNN ResNet-50 network for the defect detection task.

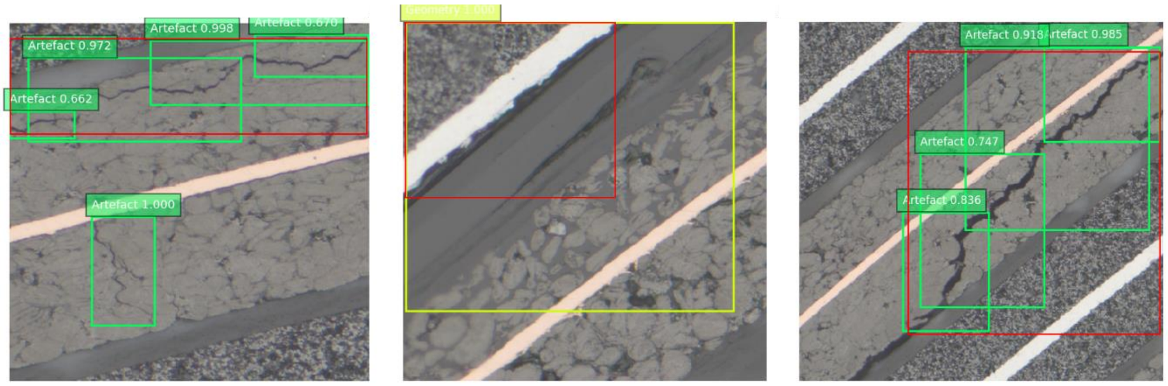


Figure 4-15: Examples where the Faster R-CNN model with ResNet-50 feature extractor correctly detected the defect but incorrectly predicted the bounding box coordinates. For instance, the model assigned several bounding boxes (green box) to the same defect in the first image, while in the second image, the assigned bounding box is much larger than the ground truth (red box). This would affect the total number of defects reported by the system.

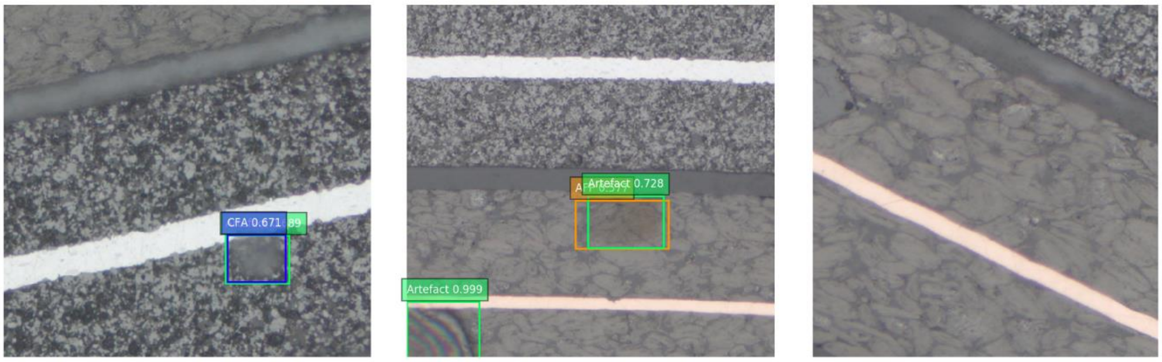


Figure 4-16: Examples where the Faster R-CNN model with ResNet-50 feature extractor makes either a false detection (left, middle) or no detection (right).

4.2 Performance assessment on full cross-section images of lithium-ion batteries

This section presents the results of the automated defect localization models for Li-ion batteries developed in Section 3.10 from the analysis of full cross-section battery images. The results based on the Faster R-CNN architecture with ResNet-50 feature extractor are compared to those of the SSD model with ResNet-50 feature extractor. Our main goal is to automate the extraction of relevant defect information from a vast amount of image data in an accurate and reproducible manner to gain valuable insights for evaluating the quality of a battery. In order to achieve this, full cross-section images are analysed with the proposed defect localization system, and the final results are presented in the form of a picture gallery of defects. In other words, if a set of battery cross-sections are fed into the system for analysis, the final output would be a gallery of images categorized according to the type of defects with their corresponding statistics. This will essentially provide a snapshot summary of the state of each battery cell, allowing for visual qualitative and quantitative assessment of the samples to facilitate various quality inspection tasks such as analysing trends in the production quality of a particular producer or comparing the production quality of various manufacturers in a much shorter time.

4.2.1 Types of defects in Li-ion batteries

In order to assess the performance of the proposed defect detection systems, a quality evaluation of commercial 18650 round cells is performed, and three different production batches are compared, as described in Table 4-6. For each batch, two cross-sections ($\sim 457\ \mathrm{mm}^2$) are obtained from each cell, and the entire cross-section is analysed using the detection models. The main focus of our analysis in this section is on seven categories of defects whose origin can be later verified through cross correlative analysis, namely: anode carbon particle (ACP), anode oxide particle (AOP), anode filled pore (AFP), anode hollow space (AHS), cathode round particle (CRP), cathode oxide particle (COP) and cathode fluoride agglomerate (CFA) as shown in Figure 4-17.

Table 4-6: Relevant information about the three commercial cells analysed for quality assessment.

LG Chem-ICR18650B4 (NMC-622-C) 2600 mAh		
Production date	Sample designation	Cross-section label
08.02.2018	INSP_044	HS18020_45, HS18020_46
27.02.2017	INSP_047	HS18020_47, HS18020_48
14.01.2017	INSP_051	HS18020_49, HS18020_50

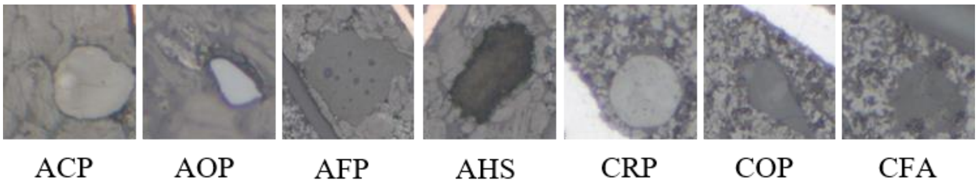


Figure 4-17: Shows an example image from each category of defect. Note that the images have been resized to have the same dimension.

4.2.2 Machine learning results of analysing full cross-section images of Li-ion batteries

Figure 4-18 shows a side-by-side comparison of the total number of defects detected in each cross-section of a battery sample by the detection systems based on both the Faster R-CNN and SSD model architectures. The highest number of defects, 335 in the cathode and 75 in the anode, were detected in cross-section HS18020_46, closely followed by cross-section HS18020_45 with 334 defects in the cathode and 70 defects in the anode, both from the same sample INSP_044. The least amount of defects was detected in cross-section HS18020_48, with 93 defects in the cathode and 48 in the anode, closely followed by cross-section HS18020_47, with 73 defects in the cathode and 74 in the anode, also from the same sample INSP_047. In general, the results show that a considerably higher number of defects were detected in the cathode layers compared with the anode layers. Furthermore, we also observed that a similar amount of defects were found in both cross-sections from the same battery sample. Lastly, the system based on Faster R-CNN detected more defects than the system based on SSD for all the analysed cross-sections.

Next, Figure 4-19 shows a breakdown of the distribution of each type of defect detected for each sample. Compared to the other categories of defects, a large proportion of the total defects were cathode fluoride agglomerates and cathode round particles, while cathode oxide particles were the least amount. More precisely, for the Faster R-CNN based system, the distribution of each type of defect is as follows: in sample INSP_044 CRP 48.9%, CFA 33.3%, AHS 8.6%, AFP 4.9%, ACP 3.2%, AOP 1.1% and no COP was detected. In sample INSP_047 the defect distribution is CFA 48.3%, AHS 22.9%, ACP 13.2%, CRP 9.4%, AFP 5.6%, AOP 0.7%, and no COP was detected. Finally, in sample INSP_051 we have CFA 42.9%, CRP 29.4%, ACP 11.7%, AHS 7.8%, AFP 4.4%, AOP 2.3% and COP 1.6%. For the system based on SSD, the distribution of defects is as follows: in sample INSP_044 CRP 44.5%, CFA 34.5%, AHS 9.9%, AFP 5.7%, 3.6%, AOP 1.3% and no COP was detected. For sample INSP_047 we have CFA 48.5%, AHS 23.4%, ACP 12.6%, CRP 8.2%, AFP 6.1%, AOP 0.9% and COP 0.4%. Lastly, for sample INSP_051 the defect distribution is CFA 43.4%, CRP 28.8%, ACP 12.2%, AHS 7.9%, AFP 4.5%, AOP 2.6% and COP 0.5%. Except for the one instance in sample INSP_047 where the system based on Faster R-CNN did not detect any COP defect and the SSD model detected 1 COP defect, the Faster R-CNN based system detected more or the same amount of defects as the SSD based system in all the other cases.

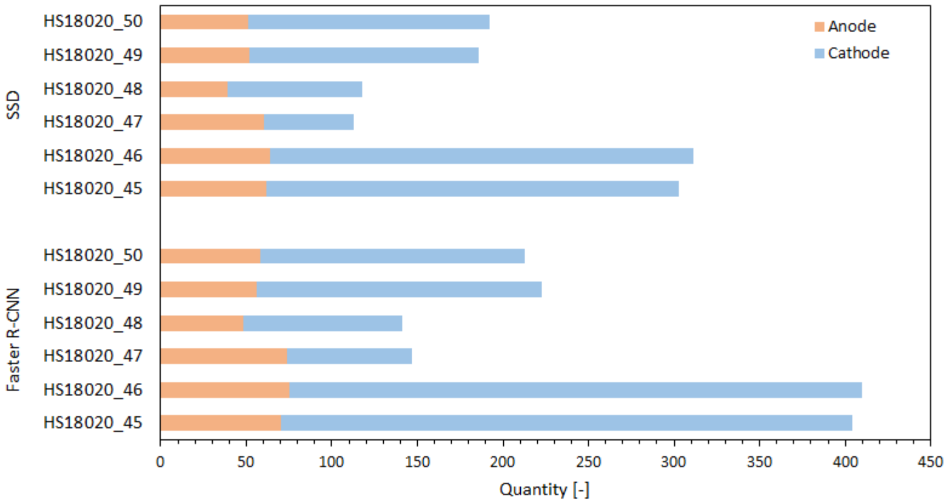


Figure 4-18: Compares the total number of defects detected in each cross-section (HS18020_45 to HS18020_50) by the Faster R-CNN with ResNet-50 and SSD with ResNet-50 based defect detection system for three different production batches.

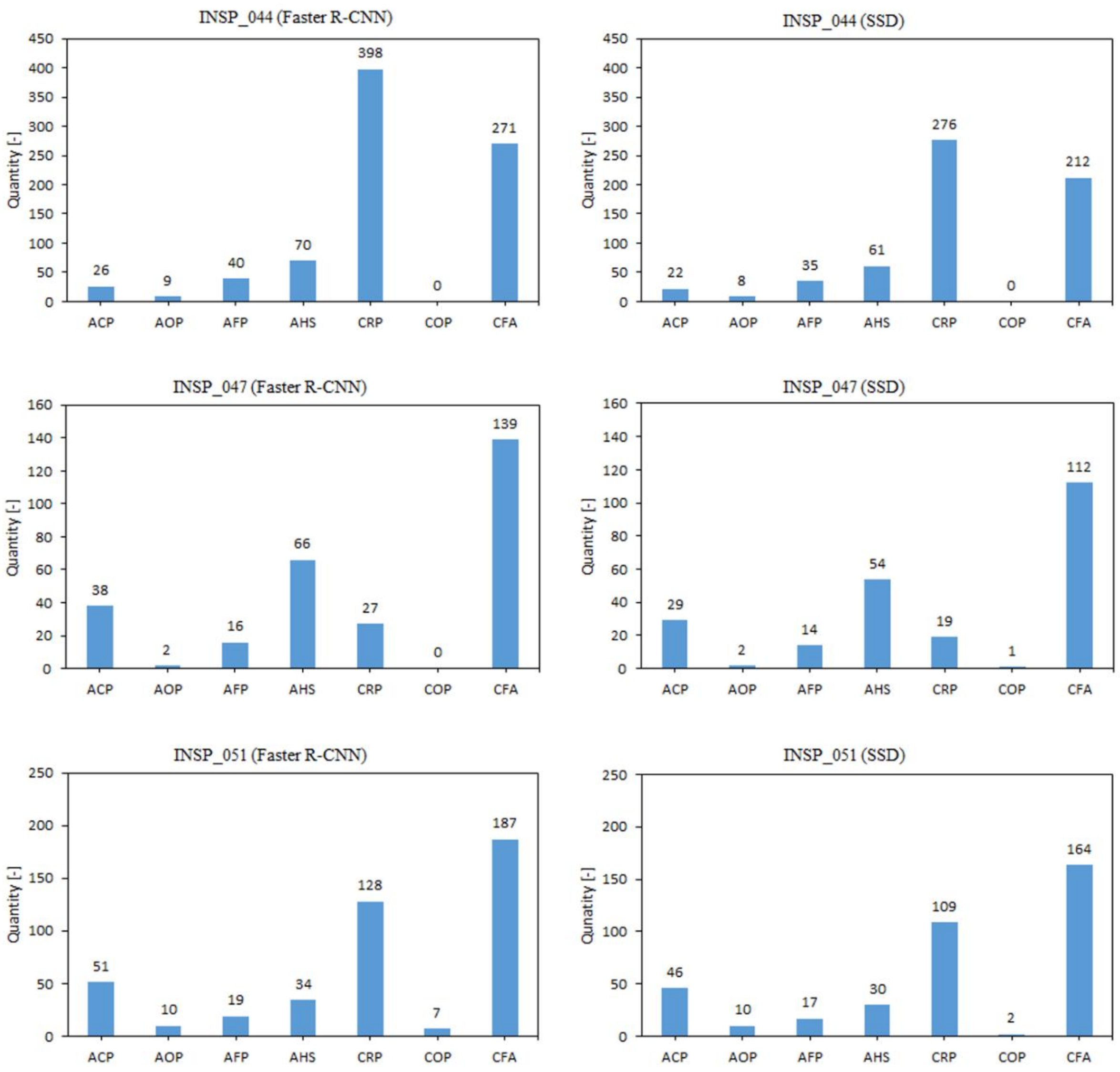
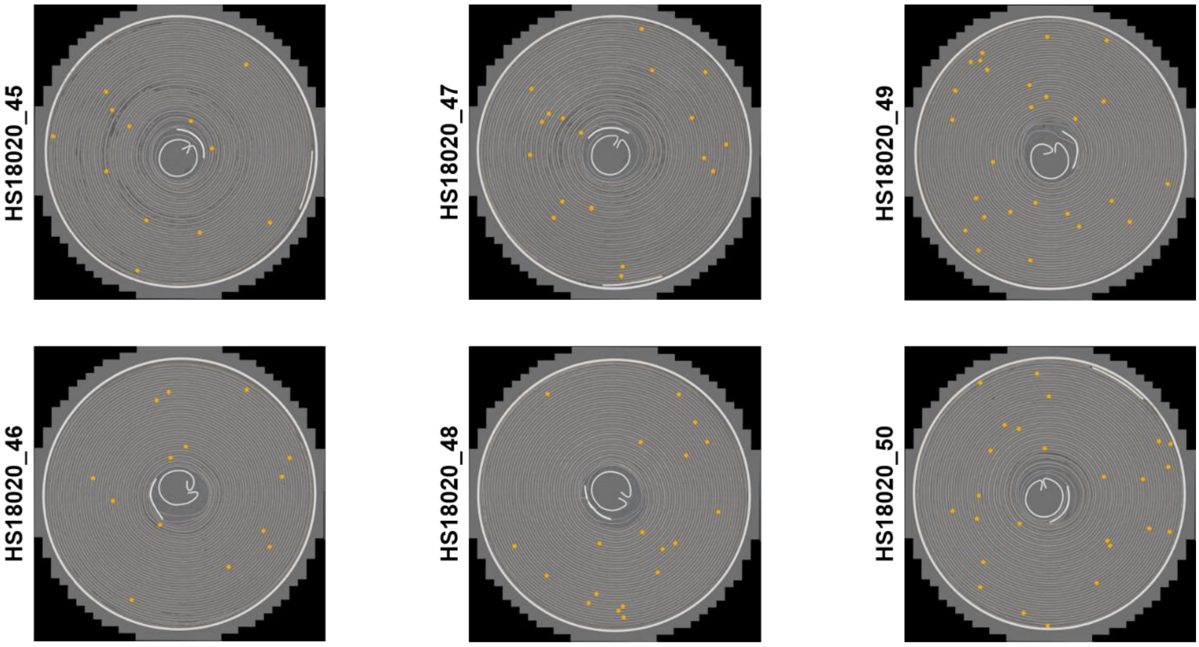


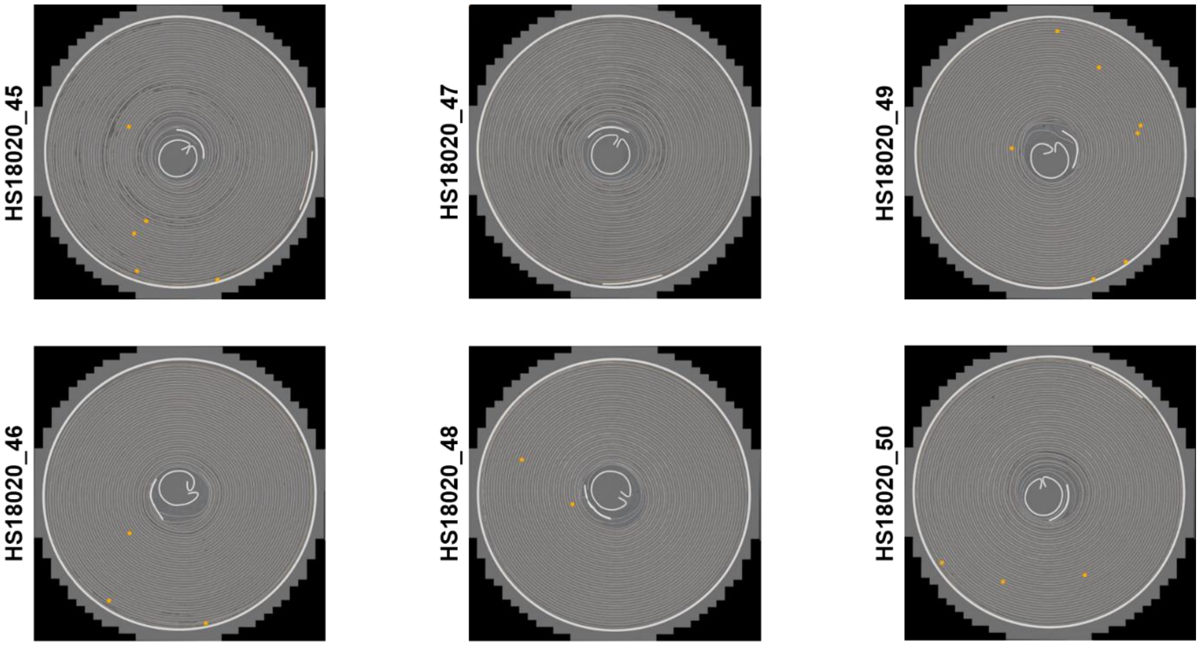
Figure 4-19: Shows the total amount of each type of defect detected in each sample for both cross-sections and compares the results for both Faster R-CNN with ResNet-50 and SSD with ResNet-50 models.

4.2.3 Spatial distribution of defects in the Li-ion battery samples

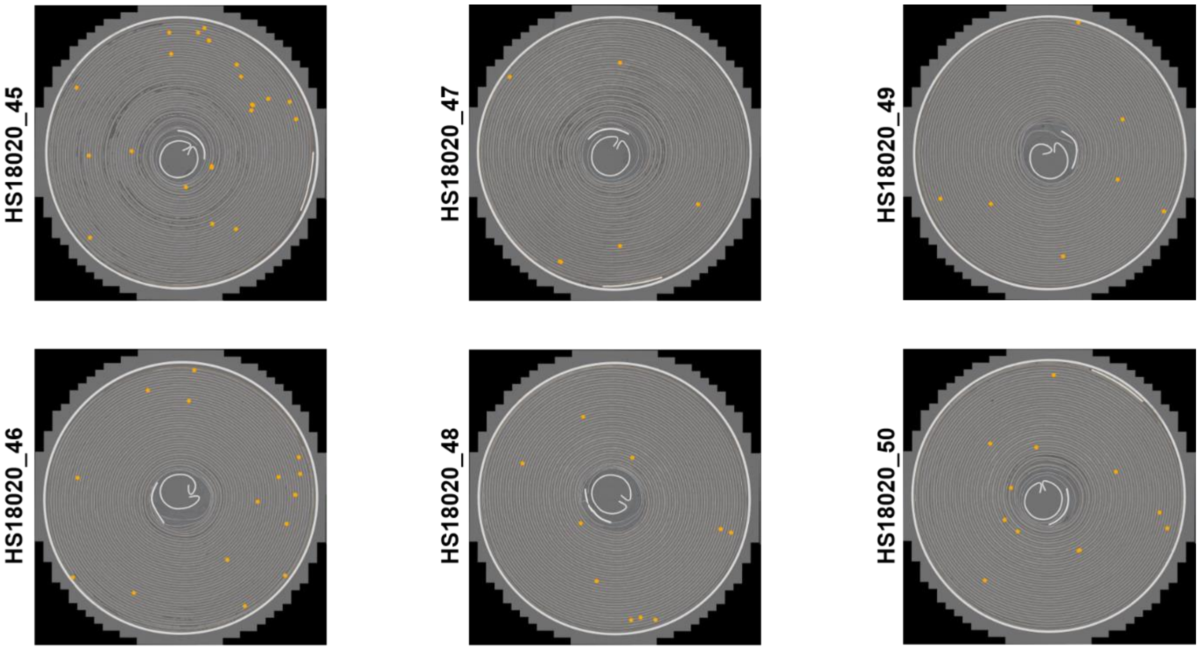
Figure 4-20 shows both cross-sections from each sample with the marked positions of the detected defects in orange dots for each defect category. This allows for further investigation of the detected defects using cross correlative analysis approaches such as using optical microscopy to obtain a high-resolution overview of the surface and then using scanning electron microscopy (SEM) with energy-dispersive X-ray spectroscopy (EDX) to verify the chemical composition of only selected regions, as we will see in Section 5.1.4. In addition, with an overview image, it is easier to discover possible trends in how defects are typically distributed given enough samples.


HS18020_45
HS18020_47
HS18020_49
HS18020_46
HS18020_48
HS18020_50

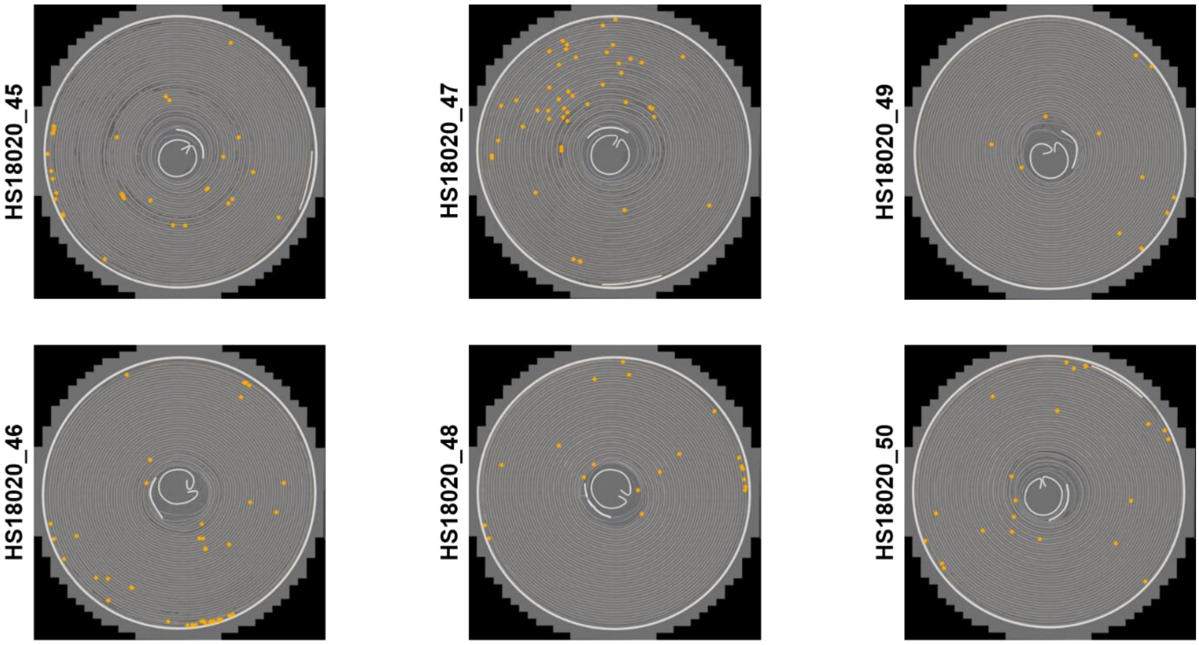
(a) ACP


HS18020_45
HS18020_47
HS18020_49
HS18020_46
HS18020_48
HS18020_50

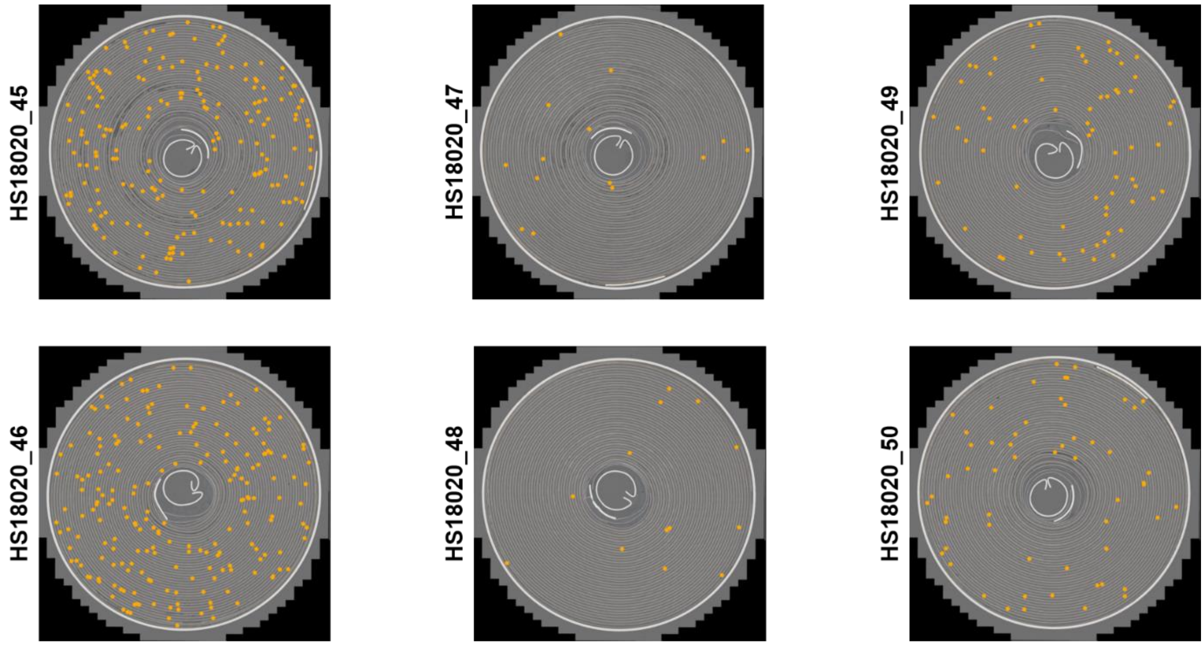
(c) AOP


HS18020_45
HS18020_47
HS18020_49
HS18020_46
HS18020_48
HS18020_50

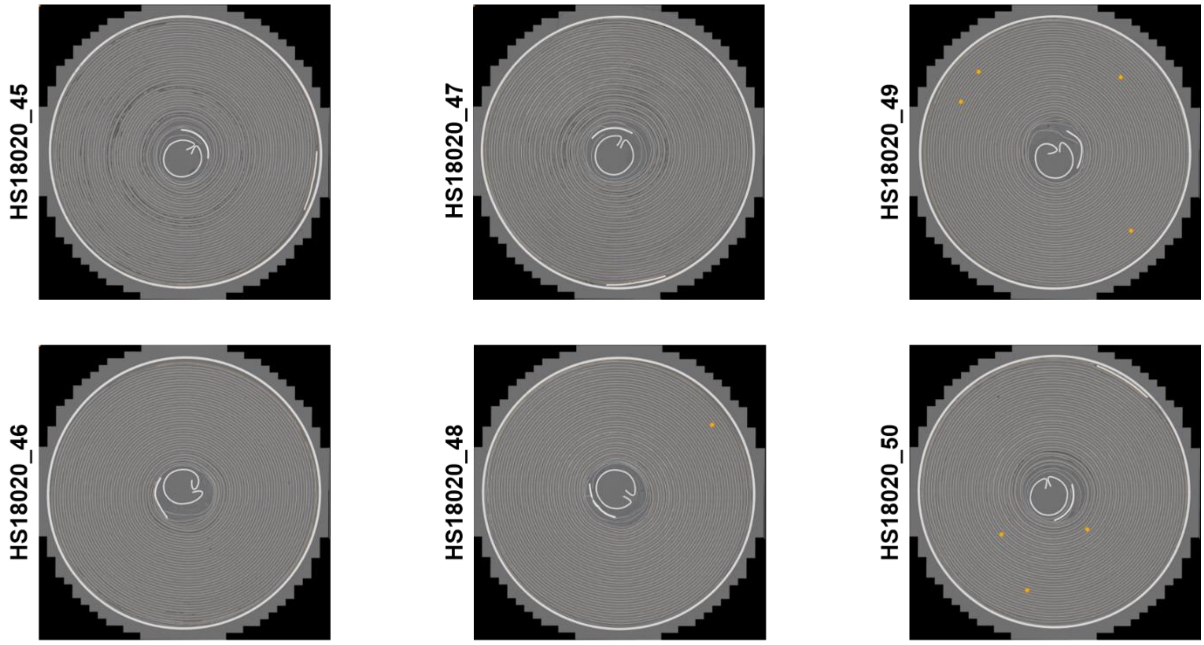
(b) AFP


HS18020_45
HS18020_47
HS18020_49
HS18020_46
HS18020_48
HS18020_50

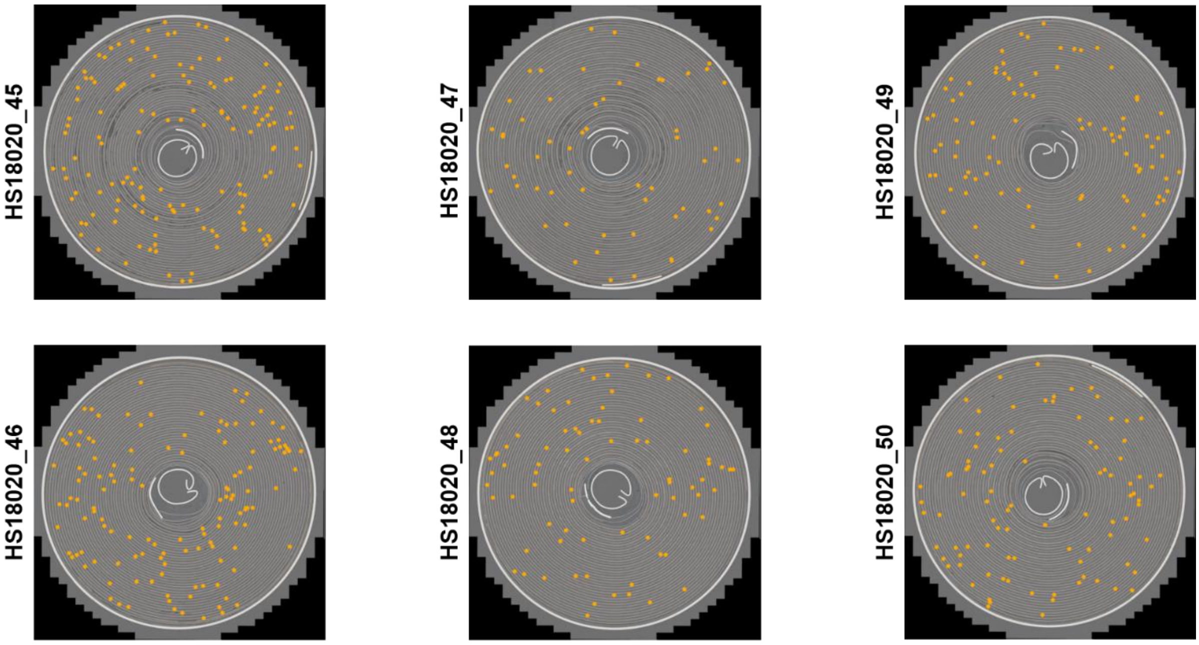
(d) AHS


HS18020_45
HS18020_47
HS18020_49
HS18020_46
HS18020_48
HS18020_50

(e) CRP


HS18020_45
HS18020_47
HS18020_49
HS18020_46
HS18020_48
HS18020_50

(f) COP

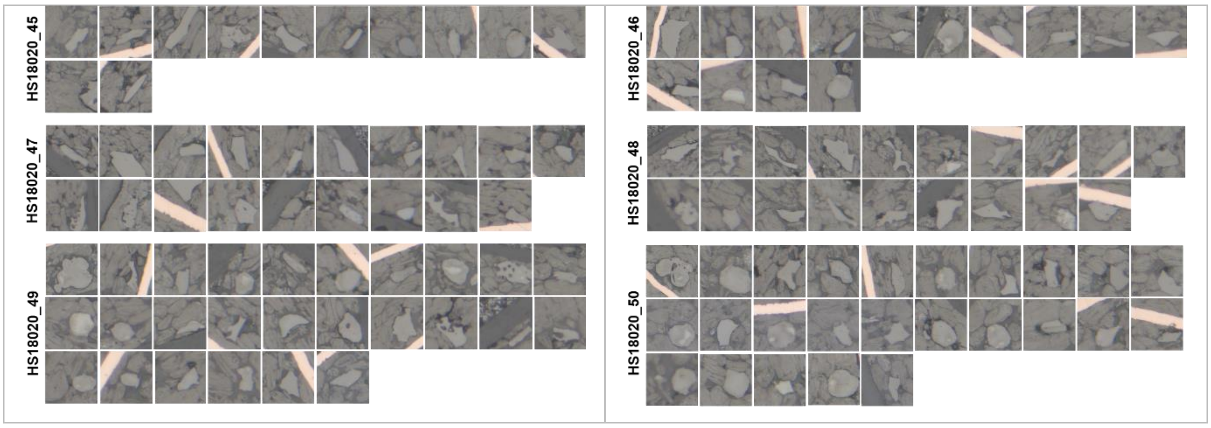


(g) CFA

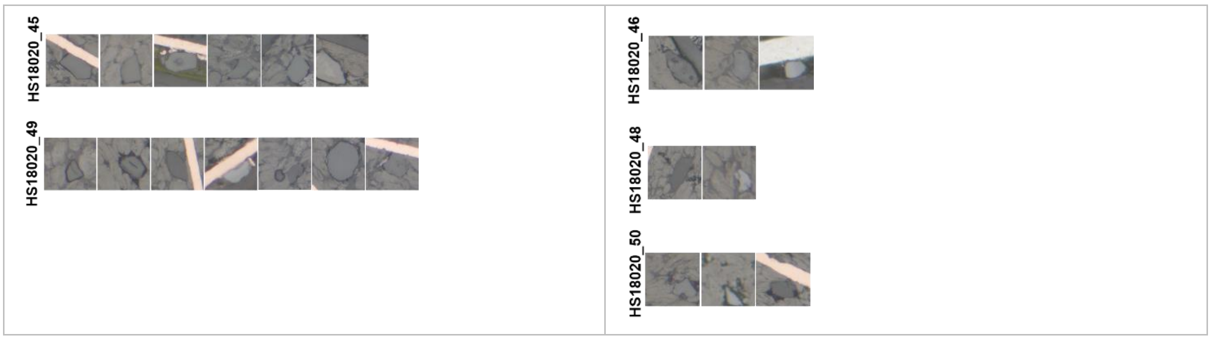
Figure 4-20: Shows the spatial distribution of detected defects for each defect category (a)-(g). The defects are marked with orange dots on the electrode coating material for each cross-section of the three production batches, INSP_044, INSP_047 and INSP_051, presented in Table 4-6.

4.2.4 Picture gallery of defects detected in the samples of Li-ion batteries by the proposed system

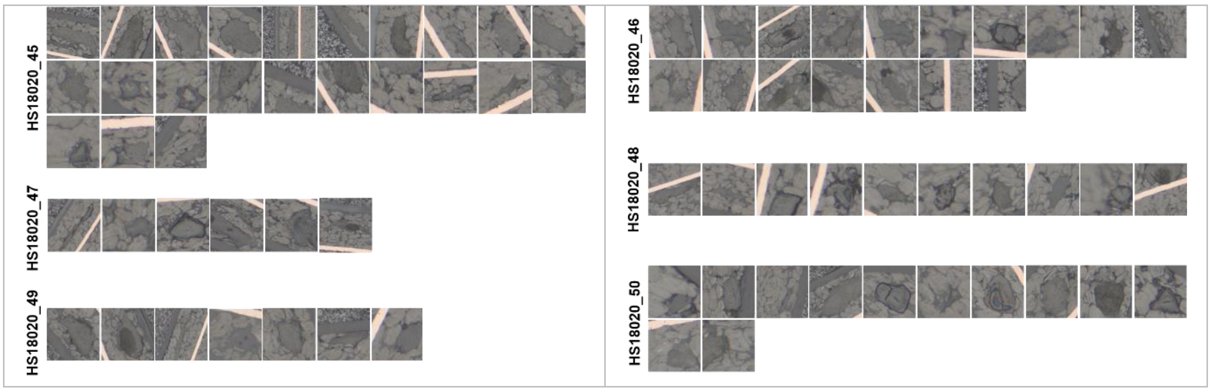
A complete picture gallery of all the detected defects from the analysed samples is presented in Figure 4-21. Note that each image in the gallery corresponds to an orange dot in Figure 4-20 above, and these images have been rescaled to the same dimension in order to save some space.



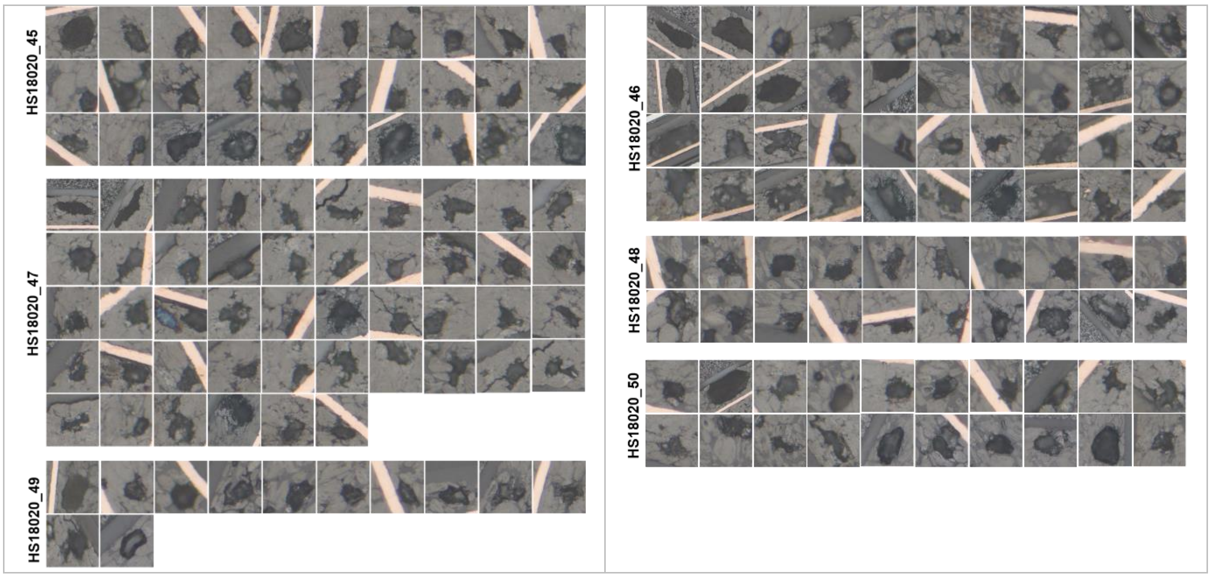
(a) ACP



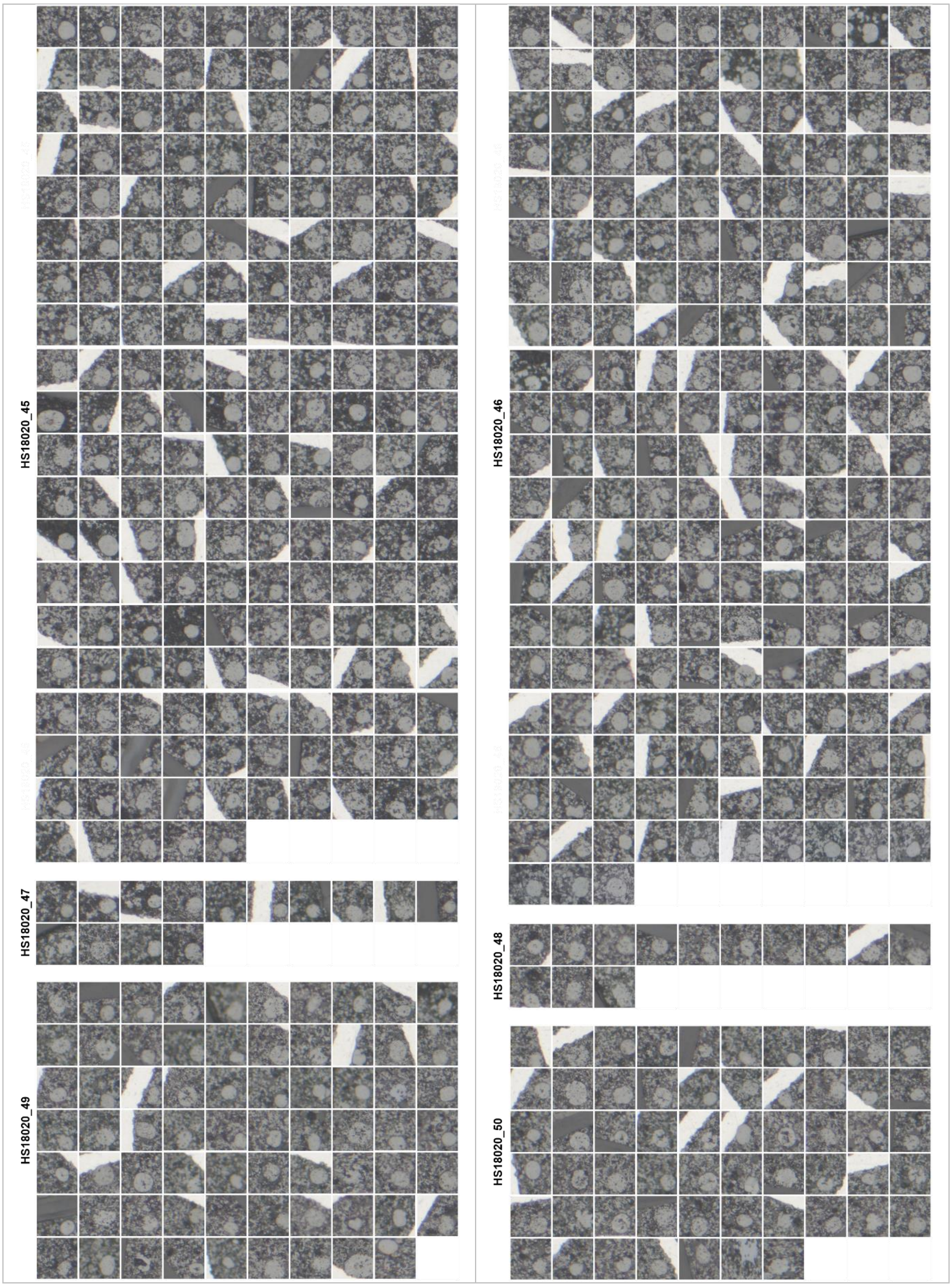
(b) AOP



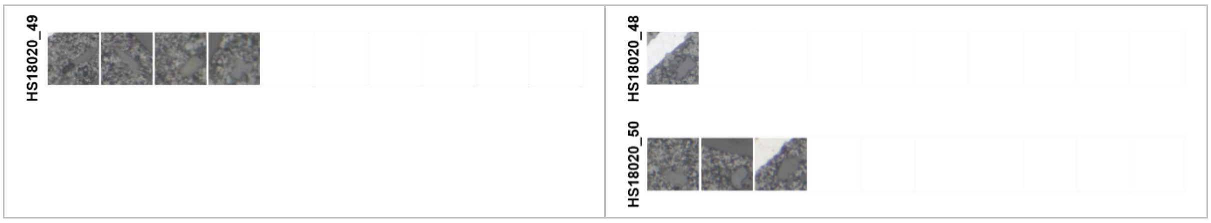
(c) AFP



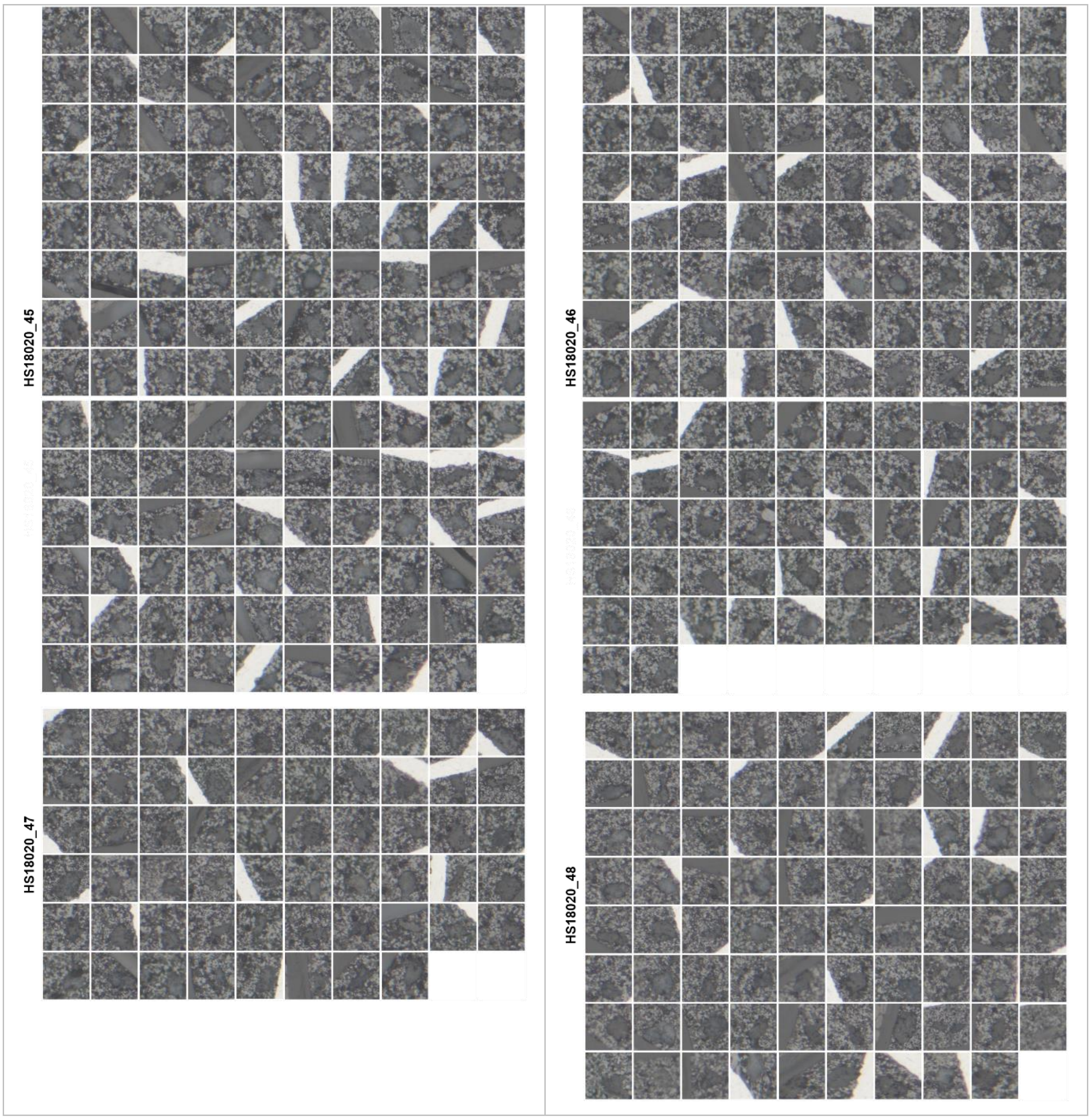
(d) AHS

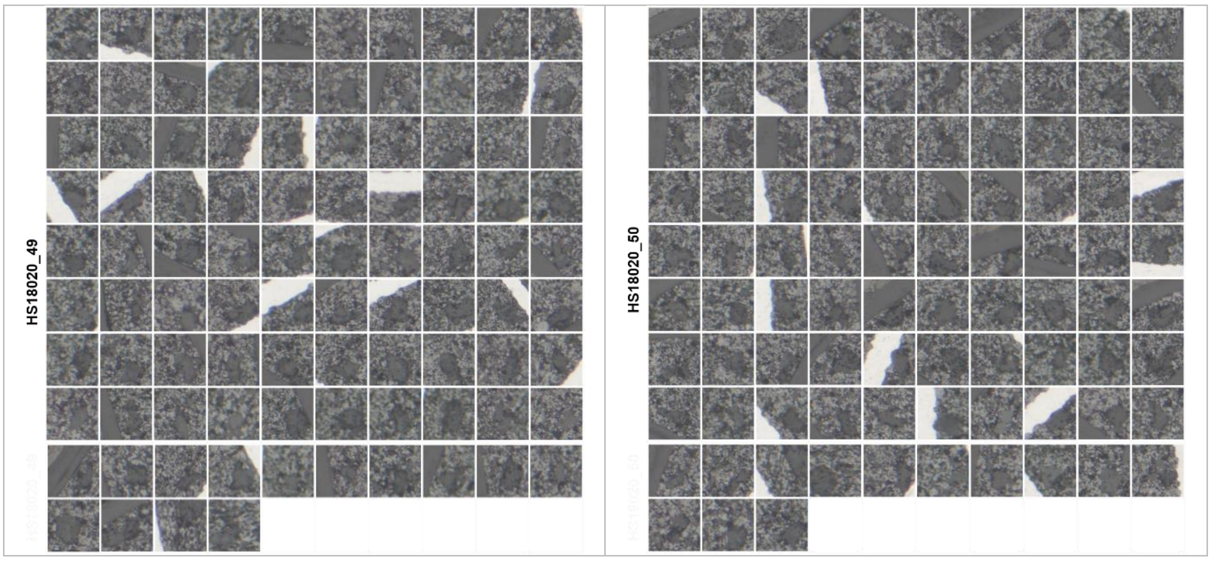

HS18020_45
HS18020_46
HS18020_47
HS18020_48
HS18020_49
HS18020_50

(e) CRP


HS18020_49
HS18020_48
HS18020_50

(f) COP


HS18020_45
HS18020_46
HS18020_47
HS18020_48



(g) CFA

Figure 4-21: Complete picture gallery of all the detected defects from each sample's cross-section. The images are cropped from the bounding box coordinates predicted by the detection system.

4.3 Experimental evaluation of unsupervised defect detection models for sintered NdFeB magnets

Similarly, we conducted several experiments while developing our unsupervised defect defection model for sintered NdFeB magnets, and the results of those experiments are presented in subsequent sections. The following section presents the results of the defect detection model based on the variational autoencoder architecture developed in Section 3.11.1. Then we present the results of the model based on the conditional generative adversarial network developed in Section 3.12.

4.3.1 Experimental results of VAE defect detection model

As discussed earlier, a common problem with unsupervised or semi-supervised approaches is the high amount of false positives produced by the model due to the lack of a well-defined target variable. Therefore, it is equally important to assess both the true and false detection to evaluate how effective the trained model is at detecting defects. In addition, the F_1-score of the model is also evaluated on the testing set. However, in order to compute these metrics, a decision threshold T_{rec} needs to be determined such that all images with SSIM $< T_{rec}$ are classified as defects. The SSIM is calculated using a sliding Gaussian window of size 11×11 with a width of 1.5. In Figure 4-22, some examples of images without any defects and the reconstructed versions are shown. The results show that the model can reconstruct most parts of the input image, albeit with some loss of fine detail. On the other hand, some examples of images with defects and the reconstructed output are shown in Figure 4-23. The model clearly finds it difficult to decode the latent vector for these images into meaningful representations, leading to a higher reconstruction error or a lower SSIM value between the input and the reconstructed output. Next, the decision threshold is determined based on the distribution of all the reconstruction errors in the training dataset and setting the value of T_{rec} as the minimum of these values, which is 0.75 in this case. Thus, any image with an SSIM less than 0.75 will be classified as a defect. For evaluation purposes, the labels of images are used but not during training. A box plot of the SSIM values for all the images in the training and testing set is shown in Figure 4-24 (a). The model can clearly reconstruct some of the images in the training set better than others, as shown by the distribution of reconstruction errors (SSIM_train) in Figure 4-24 (a). Likewise, the model can also reconstruct the normal images in the test set that were never seen during training, albeit with a slightly higher reconstruction error or lower SSIM.

On the other hand, the result shows a much lower SSIM value between images in the test set that have defects and the reconstructed output. The result also shows a higher spread between the reconstruction errors in the samples with defects than those without defects. This indicates that some samples are more irregular than others, which can be helpful in quickly finding the most severe defects when the images are presented in order of decreasing SSIM value. We explore this further in Section 4.4. Thus, with the decision threshold set at 0.75, the model can detect almost all the images with defects in the test set. However, the model also detects many false positives due to higher reconstruction errors for some images in the test set that do not contain any defects. This is a fundamental limitation of the approach because the input image is compressed by a factor of 1536 into a latent vector of size 128, which is then decoded back to the original input, resulting in some information loss. Likewise, during training, the difference between the predicted and ground truth pixels are compared using the L_2 loss which tends to produce blurry output. This is because pixel-by-pixel loss does not capture the perceptual difference and spatial correlation between two images. For example, the same image offset by a few pixels will have little visual perceptual difference for humans but could have a very high pixel-by-pixel loss. As a result, there is some loss of spatial and structural information such that only the most relevant information for reconstructing the input image in the training data is encoded in the latent vector, which therefore leads to higher reconstruction errors for unseen images without any defects but that are slightly different from the training data.

To address this issue, various values for T_{rec} were investigated to determine the optimal threshold value, which balances true and false positive detections. Figure 4-24 (b) shows the graph of true positives, false positives and false negatives for different threshold values. A threshold value of 0.69 gives the highest F_1-score of 0.70, which gives a good balance between true and false detection. A higher threshold increases the number of false positives, while a lower threshold increases the number of false negatives. In general, the optimal value for the decision threshold T_{rec} ultimately depends on the role that false negatives (i.e. missed defects) and false positives have for the task we are trying to solve. Nevertheless, given that we typically analyse thousands of image tiles, it is important to keep the number of false positives as low as possible while still accurately detecting the true positives. Therefore, in the next section, we present the results of the unsupervised defect detection model based on generative adversarial networks that can more accurately reconstruct the input image with less information loss (higher fidelity), thereby significantly reducing the reconstruction error for images without any defect.

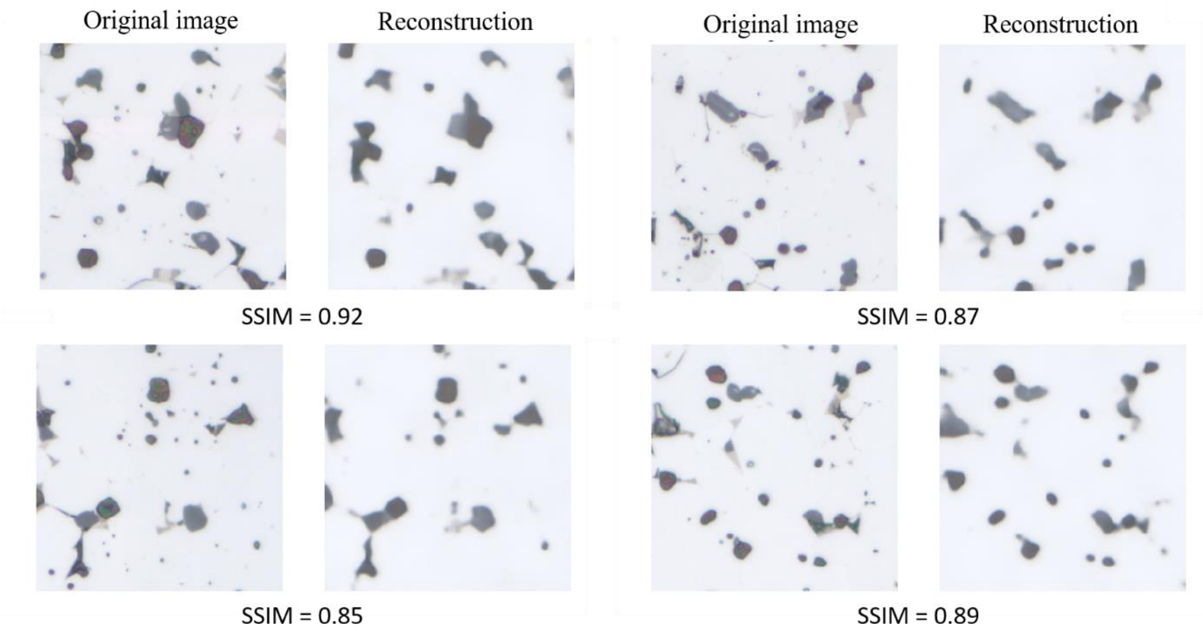


Figure 4-22: Compares the input image and the reconstructed output for examples without defects in the test set. The per-pixel loss results in blurry output.

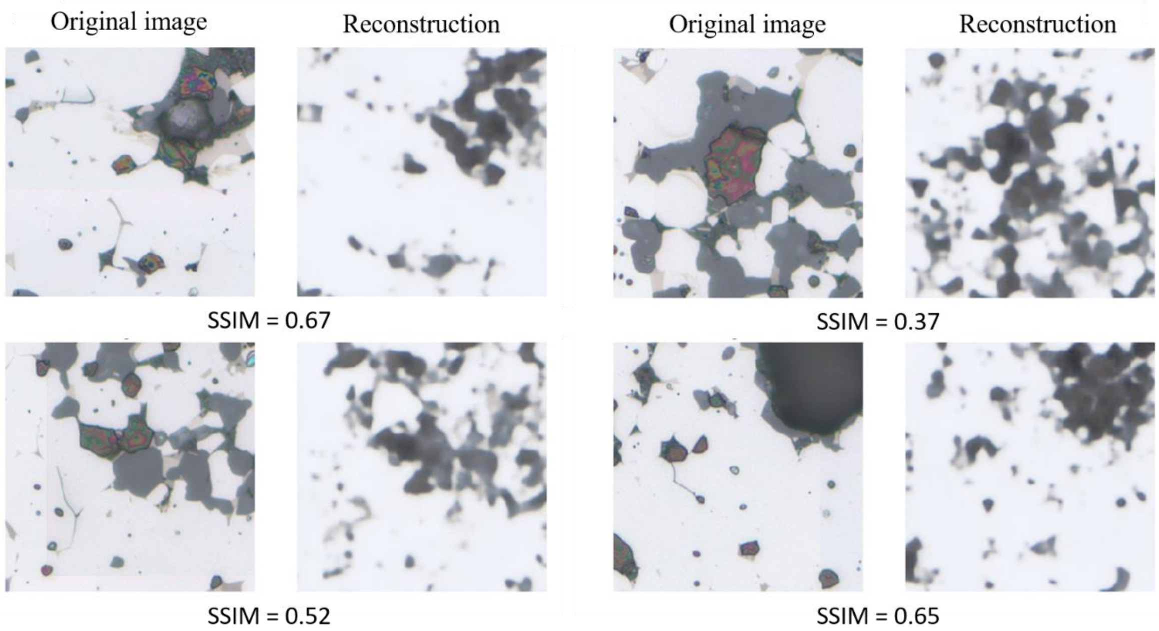
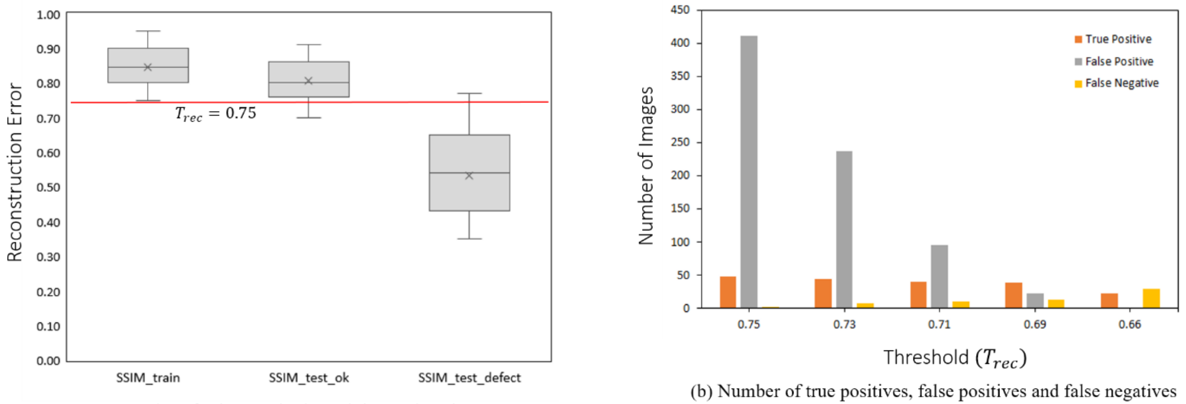


Figure 4-23 compares the input image and the reconstructed output for examples containing defects.



(a) SSIM values for images in the training and testing set

(b) Number of true positives, false positives and false negatives for various threshold values

Figure 4-24: (a) Show the reconstruction errors for images in the training and testing set. The SSIM values for the original images and the reconstructed outputs are compared for the samples in the training set (SSIM_train) and test set, which contains both normal (SSIM_test_ok) and defect (SSIM_test_defect) samples. The decision threshold T_{rec} is set to 0.75 which is the minimum SSIM value of the training set. (b) Shows the number of true positives, false positives, and false negatives for various threshold values, where a threshold value of 0.69 gives the highest F$_1$-score.

4.3.2 Experimental results of cGAN defect detection model

The model is evaluated in the same manner as during the training phase. That is, for any given image y, the edge map x is first generated by applying a Sobel filter and fed as input to the generator. Subsequently, the output of the generator $G(x)$ is compared to the original image y using the SSIM metric to determine if the image contains a defect or not based on a determined threshold value T_{rec}. Figure 4-26 shows some qualitative results of the cGAN model on randomly chosen examples from the test set that do not contain any defects. It is immediately obvious that the generator can accurately fill in the structure for a given edge map that closely matches the ground truth image. Figure 4-26 also shows the heatmap of the reconstruction errors calculated using the squared difference between the original and generated output pixels, where regions with low error values have a blue colour and regions with high residual errors are coloured red. The results show that the model is able to learn a good mapping between the edge map and the respective structure within the image such that most of the error in examples without any defect is primarily due to slight differences in pixel intensities, which is impressive considering the generator never actually observes the original image.

On the other hand, Figure 4-27 shows some examples of images with defects and the reconstructed output along with their corresponding heatmaps. Interestingly, the model can accurately reproduce every aspect of the original image except for the anomalous regions containing defects, which is precisely the desired output. Subsequently, the model is applied to all the images in the testing set, and a box plot of the SSIM values is plotted to determine if we succeeded in significantly reducing the number of false positives while still accurately detecting the images with defects. The result is shown in Figure 4-25. First, the result shows only a slight drop in the SSIM value between images in the training and testing set for examples without defects, which indicates that the model can generalize better to unseen examples than the VAE model. Secondly, the decision threshold T_{rec} is set to 0.82 which is much higher than the threshold value set for the VAE model, thus allowing for all images with defects to be detected without detecting any false positives and leaving some room for detecting even more elusive defects. Section 5.2.1 further investigates how well the model can generalize to new magnet samples whose structures differ considerably from the original training data.

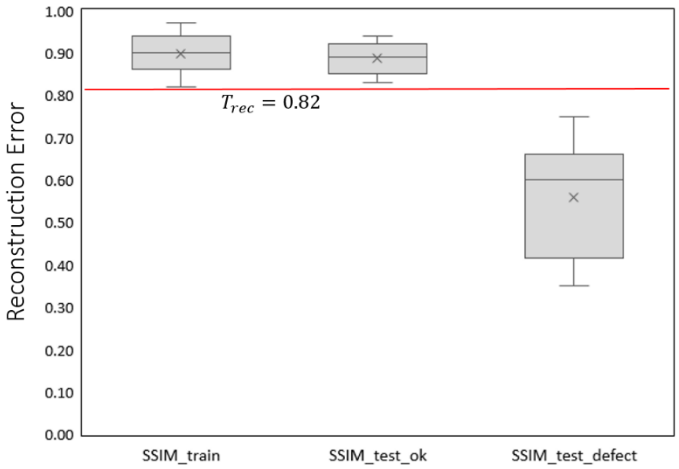


Figure 4-25: Show the reconstruction error for images in the training and testing set. The SSIM values between the original images and the reconstructed outputs are compared for the samples in the training set (SSIM_train) and test set, which contains both normal (SSIM_test_ok) and defect (SSIM_test_defect) samples. The decision threshold T_{rec} is set to 0.82 which is the minimum SSIM value of the training set.

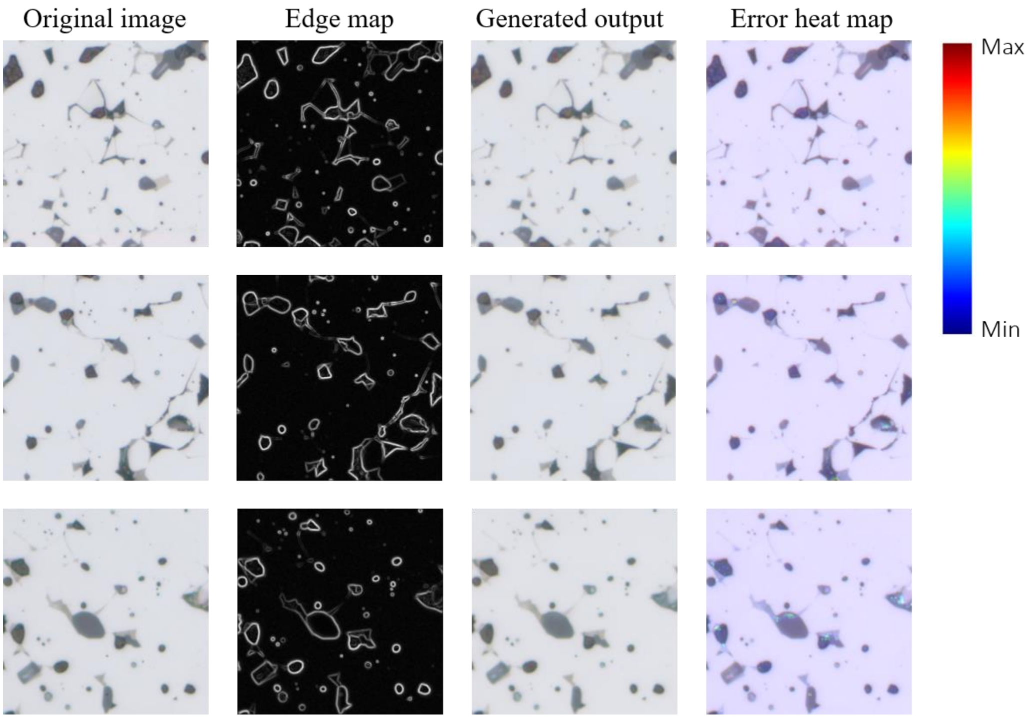


Figure 4-26: Shows examples of normal images from the test set and the corresponding output produced by the generator model. The per-pixel error value is shown in the coloured heat map, where the colour red indicates a large error and the blue colour indicates a small difference.

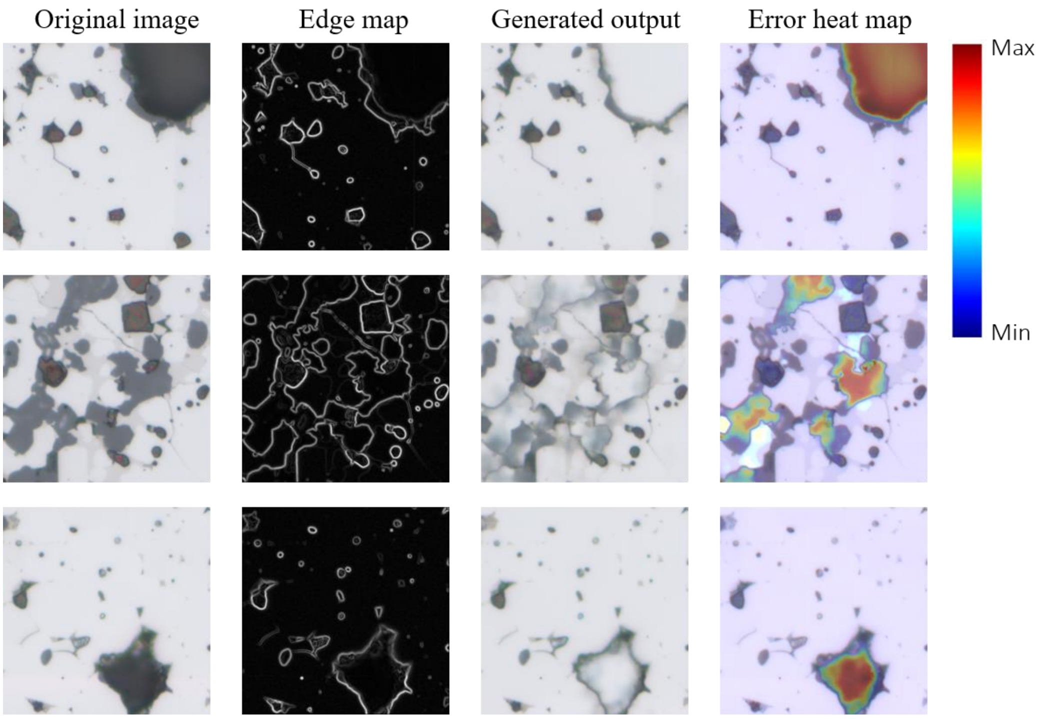


Figure 4-27: Shows examples of images from the test set which contain defects and the corresponding output produced by the generator model. The per-pixel error value is shown in

the coloured heat map where the red colour indicates a large error and the blue colour indicates a small difference.

4.4 Performance assessment on full cross-section images of sintered NdFeB magnets

This section presents the results of the unsupervised defect detection system developed in Section 3.12 from the analysis of full cross-section magnet images. Following a similar motivation for the system developed for detecting defects in Li-ion battery electrodes, our primary goal is to automate the extraction of relevant defect information in sintered NdFeB magnet microstructures. That is, extracting regions containing anomalies from very large image data to evaluate the quality of a given sample. However, compared to the system developed for detecting defects in battery samples, the model developed for analysing magnet images did not require ground truth labels. Although this typically results in a less accurate model due to the high number of false positives and the difficulty of selecting an appropriate detection threshold. Nevertheless, with an unsupervised model, there is a higher potential for discovering unknown patterns in the data. To this end, the cGAN model was applied to two magnet samples produced by different manufacturers, one with high magnetic coercivity (HS17051_90) and one with low magnetic coercivity (HS17051_35), in order to investigate the relationship between the microstructure of a magnet, in particular the number of defects and the magnetic performance.

4.4.1 Types of defects in NdFeB magnets

Figure 4-28 shows typical examples of defects in the microstructure of various NdFeB magnets, which we aim to detect automatically using the unsupervised machine learning model. This includes defects such as accumulations of oxides, large oxides, large pores, and fluorides. Moreover, it is immediately apparent that even though these magnets have similar chemical composition, their microstructures look rather different, and so do the type and structure of the defects present, which can be attributed in some extent to slightly different manufacturing conditions.

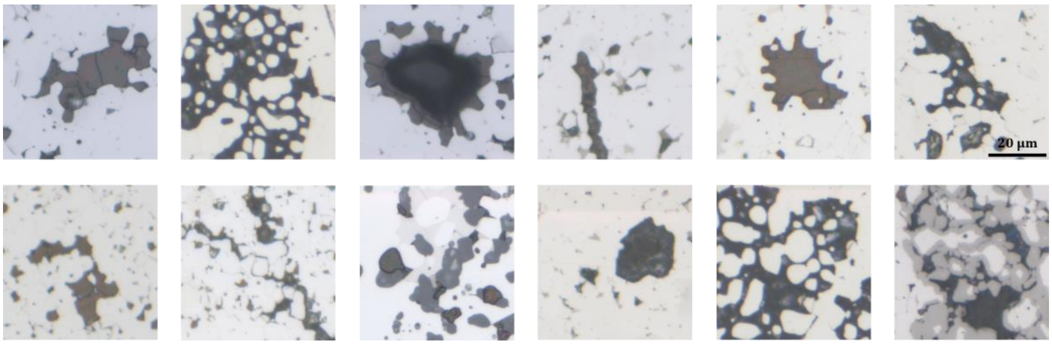


Figure 4-28: Typical examples of microstructural defects in sintered NdFeB magnets.

4.4.2 Machine learning results of analysing full cross-section images of NdFeB magnets

To examine the full cross-section of the samples, we apply the generator of our cGAN model as a sliding window classifier using a stride of size 384 (0.75 x 512). At each window location, a patch of size 512 × 512 pixels is extracted and fed as input to the defect detection system, which subsequently produces an output score of the reconstruction error. If the score exceeds the selected threshold, the patch is classified as containing a defect, and the patch and its corresponding coordinates are saved for further analysis. However, since the system can only analyse one window at a time, defects spanning multiple windows are typically split into separate sections resulting in a higher number of true positives. Nevertheless, out of 40448 patches analysed for the HS17051_90 sample, the system reported 75 patches as containing defects. However, only 21 of the 75 patches are true positives (actual defects), 37 are preparation artefacts, and 17 are false positives (see Figure 4-32). False positives refer to images that do not contain any defects or preparation artefacts but were still reported by the system. For the HS17051_35 sample, 112 patches were reported as containing defects out of 57760 total patches, of which 95 are true positives, 11 are preparation artefacts, and 6 are false positives, as shown in Figure 4-29. This means that for the HS17051_35 samples, 1 in every 608 patches contained a defect, while for the HS17051_90 sample, we have only 1 in every 1926 patches. This clearly shows that there are much fewer defects in the higher quality magnet than in the magnet with lower quality.

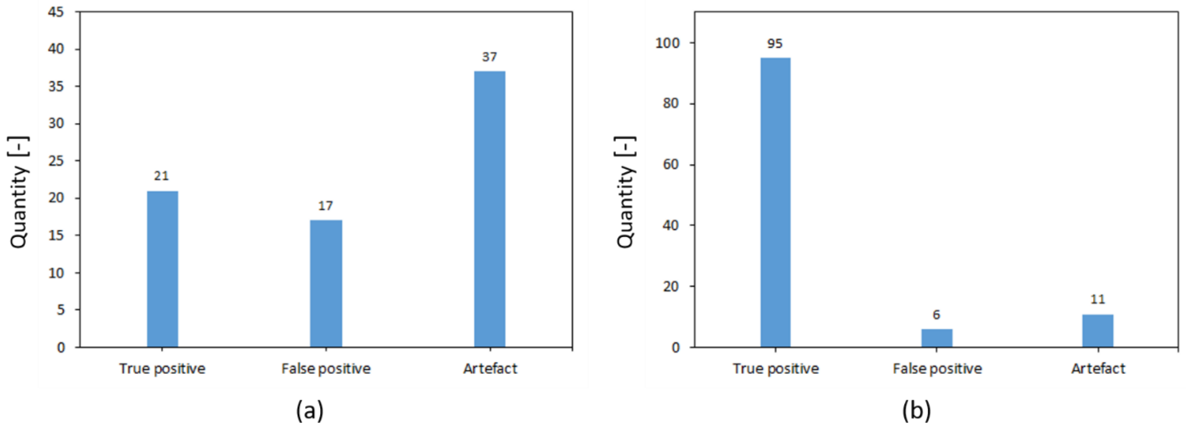


Figure 4-29: Shows the number of true positives, false positives and preparation artefacts detected for sample (a) HS17051_90 (b) HS17051_35. Note that preparation artefacts (Artefact) are manually separated from the true positives for a fairer comparison of the two samples.

4.4.3 Spatial distribution of defects in the NdFeB magnet samples

Figure 4-30 and Figure 4-31 show the spatial distribution of the detected defects enclosed in red bounding boxes and the corresponding heat map calculated from the residual error between the original and reconstructed image. A bright red spot indicates a large reconstruction error, and a dark blue indicates minimal differences. Furthermore, besides simply comparing the number of detected boxes, which can indicate how the two samples compare, the heatmap also provides another means to evaluate the samples by providing a quick overview of the severity of the defects indicated by the distribution of bright spots in the heatmap.

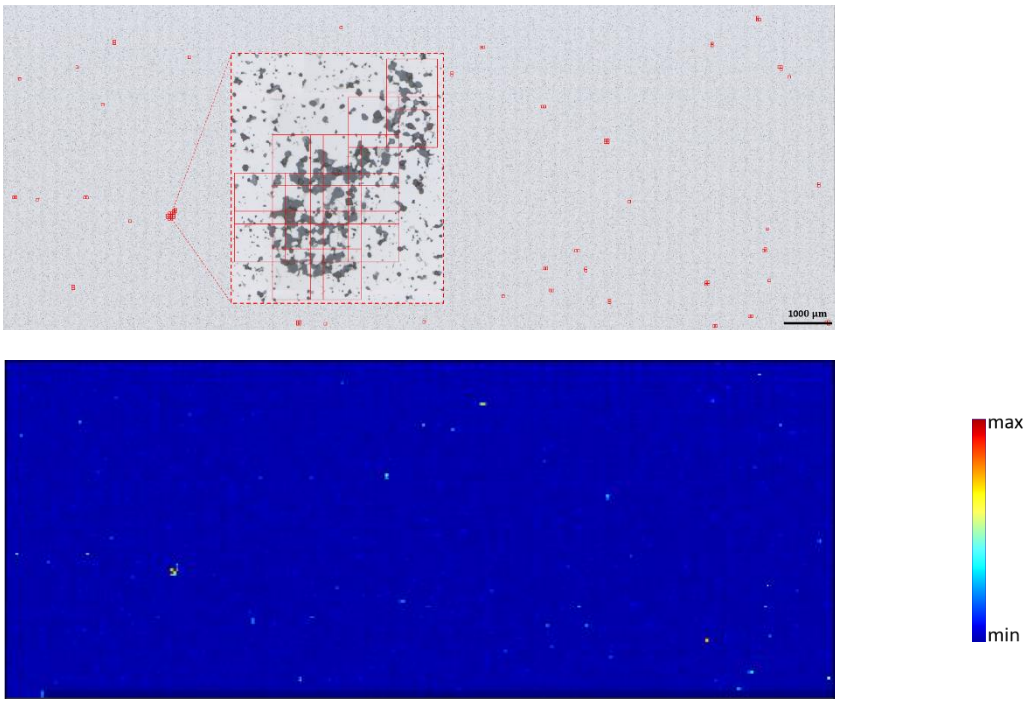


Figure 4-30: Shows the spatial distribution of detected defects in red bounding boxes and the corresponding heatmap for the sample HS17051_90.

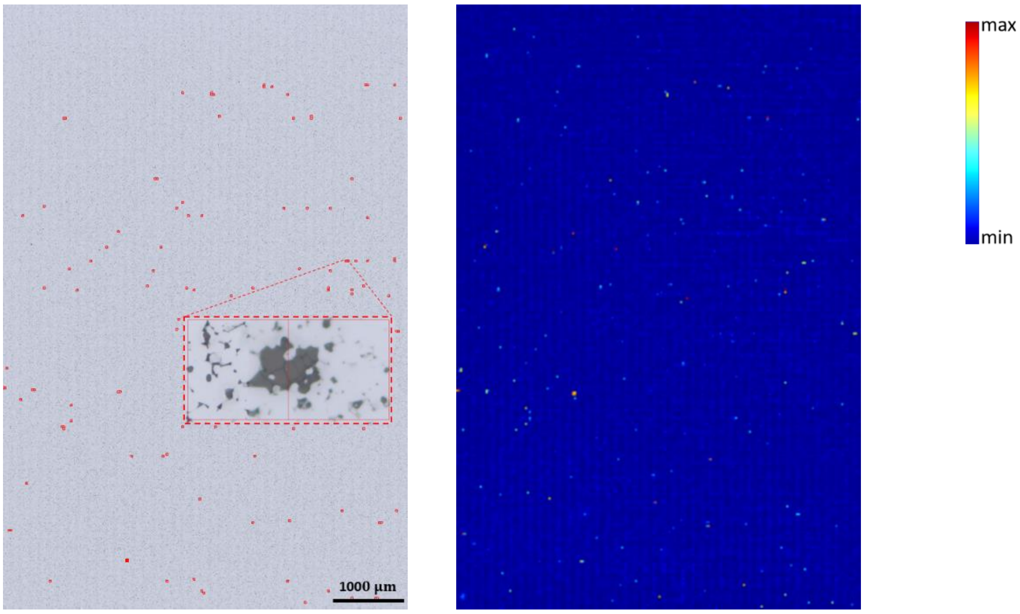


Figure 4-31: Shows the spatial distribution of detected defects in red bounding boxes and the corresponding heatmap for the sample HS17051_35.

4.4.4 Picture gallery of defects detected in the samples of NdFeB magnets by the proposed system

Figure 4-32 presents a picture gallery of all the detected regions in the two examined magnet samples. The images are cropped from the bounding box coordinates predicted by the detection system. Note that the images are ordered according to their reconstruction errors from highest in the top left to lowest in the bottom right. However, since the results have several preparation artefacts and false positives, Figure 4-33 shows only the true positives after manually removing the false detection. Nevertheless, the number of detected defects in the low coercivity magnet HS17051_35 is much higher than in the high coercivity magnet HS17051_90.

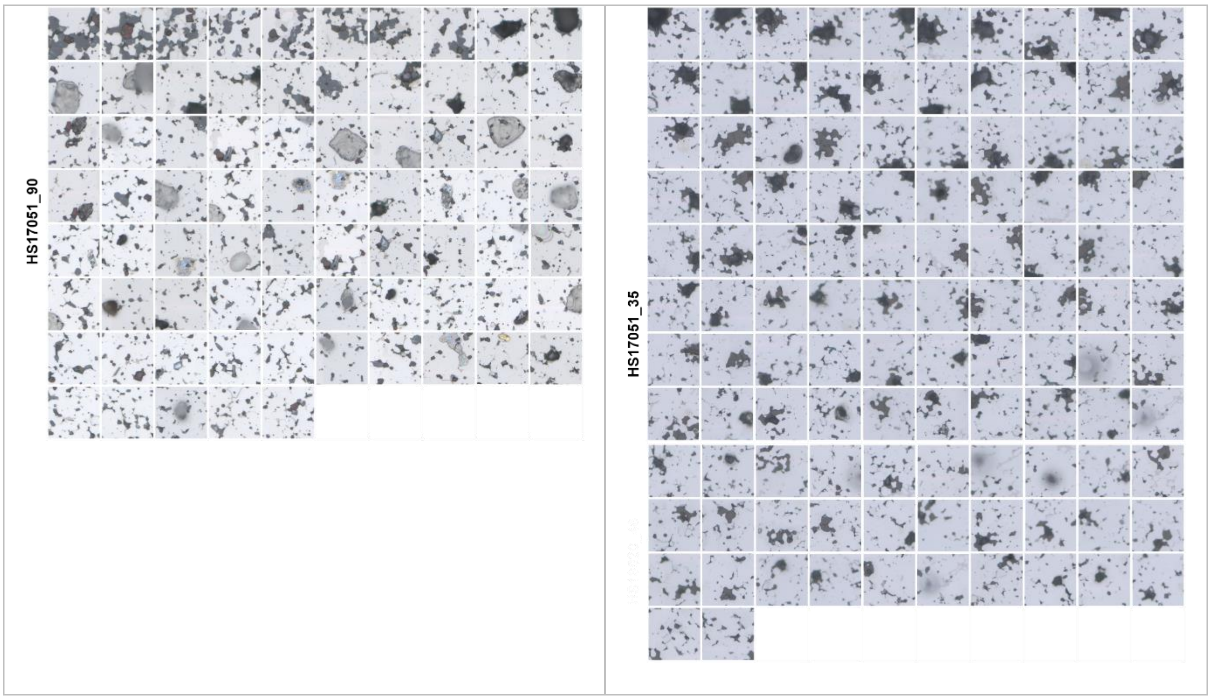
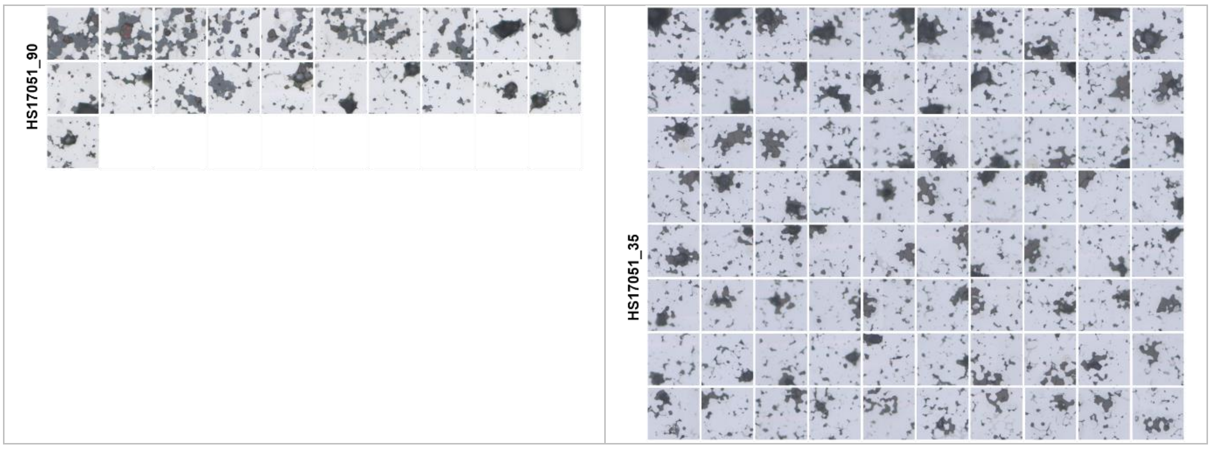


Figure 4-32: Picture gallery of all the detected regions in the HS17051_90 and HS17051_35 magnet samples before removing the false positives and preparation artefacts.



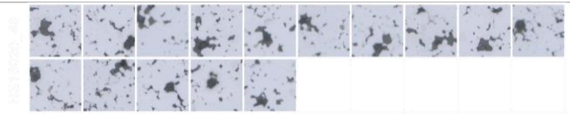
	

Figure 4-33: Picture gallery of all the detected regions in the HS17051_90 and HS17051_35 magnet samples after manually removing the false positives and preparation artefacts.

Chapter 5

Discussion

5.1 Detecting defects in lithium-ion batteries

During the development of our proposed defect detection system for Li-ion batteries, several experiments were conducted to better understand the system's performance under various conditions. This chapter discusses the results of those experiments and explores the various properties of our proposed system. First, the performance of the underlying deep learning model will be evaluated, and the impact of factors such as network depth, data requirements, classification versus object localisation, single-stage versus two-stage detection and runtime evaluation will be discussed. In subsequent sections, the performance of our proposed system will be compared to manual visual inspection in addition to classical image processing methods. Then the relevance and benefits of the proposed system will be discussed, and lastly, some drawbacks and areas for further improvements will be considered.

5.1.1 Performance assessment of the proposed methods

Chapter 3 explored various model architectures, from simple image classifiers using a sliding widow detector to more complex object detection and localisation models. Various methods were investigated to cope with the highly imbalanced data set using under-sampling, over-sampling and weighted loss functions. Nevertheless, some models still performed better than others. Therefore, in this section, we will look at certain aspects of the underlying defect detection problem in relation to the model architectures and examine which aspects have the most influence on the overall performance.

Network depth

Deep convolutional networks have been mainly responsible for the recent breakthroughs in image recognition tasks in the last decade. A lot of evidence [152], [169]–[171] has shown that network depth is of utmost importance for achieving top performance, and the top results on the ImageNet dataset all exploit very deep models. Most of the problems which in the past plagued the training of deep networks and prevented the models from converging have been largely mitigated by using better activation functions in intermediate layers and with normalized initialization [156], [170], [172], [173] and intermediate normalization layers (batch norm). Therefore, it is relevant to ask whether we can achieve better performance on our defect detection task simply by using a deeper network. For VGG-style networks where each layer is directly connected to the next layer, it has been experimentally shown in [174],

[175] that for a given task, there is an optimal number of layers, after which accuracy gets saturated, and the training and test error starts to increase. This is called the *degradation problem*. However, note that this is not the result of overfitting since adding more layers also causes the training error to increase.

On the other hand, in residual networks (ResNet), where some of the layers do not directly operate on the input from the previous layer alone, instead, the input is passed on unchanged for several layers using shortcut connections, whereby the following layers operate on the residual of the passed input and the output of the previous residual layer. This is particularly relevant for our defect detection tasks since most of the categories of defects we aim to detect often occupy only a very small area of the whole image. Furthermore, the texture and appearance of these defects are typically very similar to the rest of the active material, which makes the problem even more challenging. Therefore, considering that a much smaller number of pixels are available to represent the information of these small defects, which is also much fewer than the non-defective regions, this means that there is less representative information for the detector to operate on. Besides, key features of the much smaller area containing defects are vulnerable and even progressively lost when going through various kinds of layers in the deep networks, such as convolutional or pooling layers. For example, in the VGG-19 network, if the defect occupies an area of 32×32 pixels, after going through 5 pooling layers, it will be represented by at most 1 pixel. Whereas, with the use of shortcut connections in the ResNet-50 network, most of the relevant features can simply be shuffled across without any loss of information.

In a subsequent experiment, we trained a ResNet-101 network which has 101 layers, in order to investigate whether an even deeper model will lead to a performance boost. However, we did not observe any significant performance improvement compared to the ResNet-50 network, which could be due to the relatively small size of the dataset, such that the ResNet-50 model is already able to learn most of the relevant features from the data and the addition of more layers will only increase the chances of overfitting.

Evaluation of the multiclass classifier models

In Section 4.1.2, the results of our experiments while developing the multiclass classification models for detecting defects in Li-ion batteries were presented. From the results, we observed the same performance between the ResNet-50 model trained with a weighted cross-entropy loss function and the model trained with the oversampling technique, which was surprisingly below the performance of the model trained with the standard cross-entropy loss function. This is probably because the weight terms might have slightly biased the model towards the minority class, while the over-sampling technique suffered from overfitting, as shown in Figure 4-2. Thus, both techniques did not improve the model performance compared with the baseline cross-entropy loss function. However, for the VGG-19 model, modifying the loss function and balancing the data helped improve the performance of the models compared with using standard cross-entropy loss, although the performance is still much lower than the ResNet-50 models.

Next, the AUROC and AURPC of the top-performing models are discussed. Figure 4-3 shows that both ResNet-50 models trained with focal loss and standard cross-entropy achieve similar AUROC. However, the model trained with focal loss achieved a higher AUPRC for all classes except the geometric irregularity class than those achieved with a standard cross-entropy loss function (see Figure 4-4). This indicates that the model with focal loss was more confident in its class predictions, as shown by the high precision and recall for various threshold values. The confusion matrices for the ResNet-50 model are presented in Figure 4-5, and those for the VGG-19 model are presented in Figure 4-6. First, note that both the ResNet-50 and VGG-19 models achieved precision and recall greater than 90% for all the defect classes. The ResNet-50 model trained with standard cross-entropy misclassified 6 examples containing foreign particles as no defect class compared to only 2 misclassified examples for the same model trained with focal loss. In addition, 5 images with no defect were misclassified as preparation artefacts compared to 3 images for the model trained with focal loss. This could indicate that the model trained with standard cross-entropy was slightly biased toward the majority class in both cases.

From the confusion matrices in Figure 4-5, there is no strong evidence that the model had difficulty classifying any particular class when comparing the model trained with standard cross-entropy with those trained to deal with class imbalance. This is probably because the class imbalance between the majority and minority classes is not severe. However, the results show an improvement in the model performance when using the focal loss function. The validation losses of both models using all four training methods are compared in Figure 4-2. The results show that the networks trained with focal loss function converged to the minimum much faster than the other training methods. Lastly, the ResNet-50 model trained with standard cross-entropy achieved a better performance than all the VGG-19 models.

Classification vs object detection model

Ultimately, our goal is to analyse very large images to detect and quantify the number of defects present in a given cross-section. Therefore, in our initial attempt, a multiclass classification model was trained to detect defects and then applied to full cross-section images using a sliding window approach. This was done as follows: at each sliding window location, an image patch is extracted, scaled and fed through the CNN, and then the class with the highest probability is selected before sliding the window to the next location and repeating the process. However, since defects may appear in various sizes, the images also need to be processed at different scales. Furthermore, to avoid misclassifying objects at the image border, image patches will have a lot of overlap. As a result, the CNN will repeatedly compute the same features, making the entire process extremely slow. Then NMS is applied to reduce the amount of non-informative, duplicate detection, which is characteristic of this strategy. In addition, for a given image patch containing more than one defect category, the model will simply assign the patch to any one of the defect categories present, eventually leading to underestimating the defect population. Moreover, given that the output of the classification model is simply the predicted class probability, there is no easy way to determine the exact location of the defect in the image without resorting to methods such as extracting regions from CAM heatmaps, which can become very cumbersome. Furthermore, the CAM approach can only detect defects from the same class in one image; it cannot detect defects from different classes. As a result, it is not feasible to perform correlative analysis with this approach. Although this is a reasonable first attempt at defect detection, it has several flaws:

- It is relatively slow, making it impractical for processing very large images.
- The accuracy of the bounding box depends heavily on the parameters selected for the sliding window, the image pyramid and the ROI size.
- It is impossible to train the entire architecture end-to-end, which means that errors in bounding-box predictions are not backpropagated through the network to produce better, more accurate detections in subsequent iterations through weight updates. Instead, the model weights are fixed and only used for prediction.

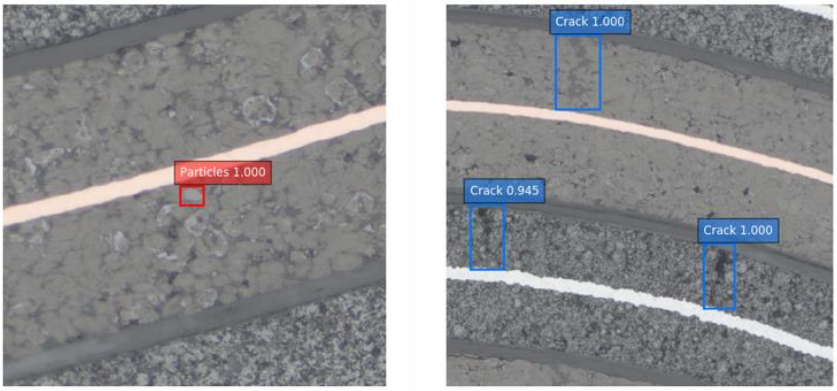
Nevertheless, this demonstrates the effectiveness of the proposed approach for detecting various defects in the electrodes of LIBs. Furthermore, the network needs to perform well on image classification tasks to achieve high performance on localization as it involves identifying both the object category and accurately predicting the bounding box coordinates. Subsequently, we investigated end-to-end trainable object detectors, which can classify and localize many objects at varying scales in a single forward pass. We developed two defect localization models based on one-stage and two-stage approaches, which can detect multiple defects of different sizes and place tight-fitting bounding boxes at each defect location in the same field of view in near real-time.

One-stage vs two-stage detection

Many detection methods based on deep networks have been proposed to tackle object detection problems in the past years. They typically suffer from the speed-accuracy trade-off and can generally be categorized into one-stage or two-stage detectors. Most of these state-of-the-art detectors typically have some difficulty detecting small objects [176]. Therefore, in our experiments, we investigated both one-stage and two-stage detectors, namely SSD and Faster R-CNN, to detect and localise microstructure defects. Building on the results of the multiclass classification model, we evaluated our defect detection models using both ResNet and VGG backbones to determine how much they affect the model's overall performance. In general, from our experiments, Faster R-CNN achieved a better performance than SSD, particularly on the defects with smaller areas irrespective of the backbone feature extraction network, which is consistent with the findings in [176], [177]. Figure 5-1 shows the predicted scores of both the Faster R-CNN and SSD models on the same set of images. For each defect category, the predicted score or confidence of the Faster R-CNN network is typically higher than those of the SSD network. This can be mainly attributed to two factors: 1) the region proposal network (RPN), which generates high-quality region proposals, and avoids the need to generate an excessive number of proposal boxes. This reduces the number of false positives and improves accuracy, mostly leaving the bounding box refinement and classification to the Fast R-CNN network. On the other hand, SSD eliminates the need for the proposal network to enhance the runtime speed, which consequently results in a drop in detection accuracy. However, this is compensated for by generating many default boxes and making predictions on multiscale feature maps, using earlier layers to detect small objects and lower resolution layers to detect larger objects. This approach is faster than using a separate network to generate region proposals because computations of the convolutional layers are effectively utilized to create a feature pyramid from a single image scale. However, the problem with this is that the low-level activation maps of the convolutional network contain less semantic information than the high-level activation maps. Consequently, the high-resolution scales of the feature pyramid contain less relevant information for detection than lower resolution scales of the pyramid. 2) One-stage methods such as SSD use a soft sampling method that uses all the information in each batch to update the network parameter rather than selecting samples from the training set. Two-stage detectors, on the other hand, such as Faster R-CNN, employ hard sampling methods that randomly sample a certain number of positive and negative bounding boxes and try to balance the ratio between the two sets to update the network parameters. Therefore, considering there is a high imbalance between defective and non-defective regions, this makes a significant difference in the model performance since the one-stage detector is more likely to misclassify defects with a similar appearance to the more dominant background class.

Next, when comparing the backbone networks, the models with ResNet-50 backbone outperforms the one based on VGG-19 in both one-stage and two-stage detection networks. This is unsurprising since the ResNet-50 also performed better on the classification task. Other approaches, such as SOD-MTGAN [178], have been proposed to detect small objects by taking cropped regions as input. However, they also rely on pre-processing steps obtained from a baseline detector such as Faster R-CNN. An alternative approach that was also investigated is based on a system called Feature Pyramid Network (FPN) [179], which proposes a solution to detecting objects at different scales by performing a top-down computation step after generating the feature maps in a bottom-up computation. In order to achieve this, the top-down pathway generates higher resolution features by upsampling spatially coarser but semantically more interesting high-level low resolution activation maps from higher pyramid levels and summing them with low-level high resolution features from the bottom-up pathway through the lateral connections. The idea is to combine low level activations which are more accurately localized since they have been subsampled fewer times with higher level features maps to detect objects at different scales with improved precision and at marginal extra computation time. However, in our experiments we found that they provided a slight improvement in precision (mAP 0.85) compared to the baseline of one scale, but the precision-time (0.63 second per image on GPU) trade-off was not advantageous.

Faster R-CNN



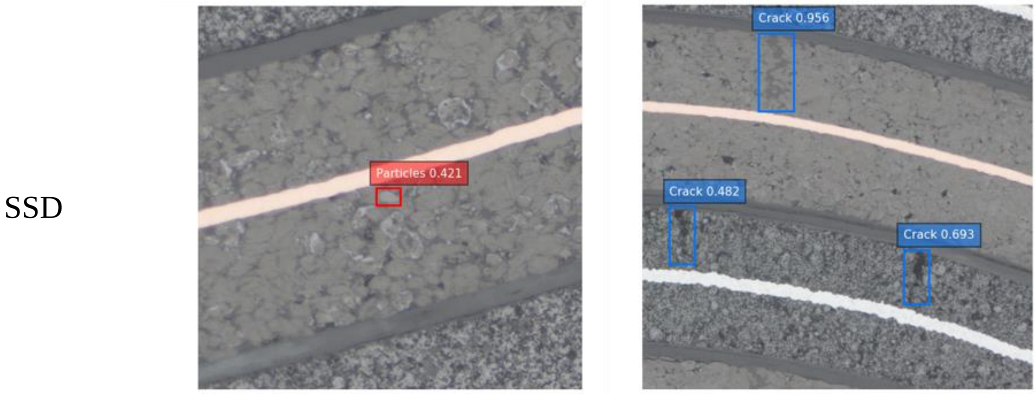


Figure 5-1: Shows the prediction confidence for Faster R-CNN with ResNet-50 and SSD with ResNet-50 on the same set of images.

Data requirements

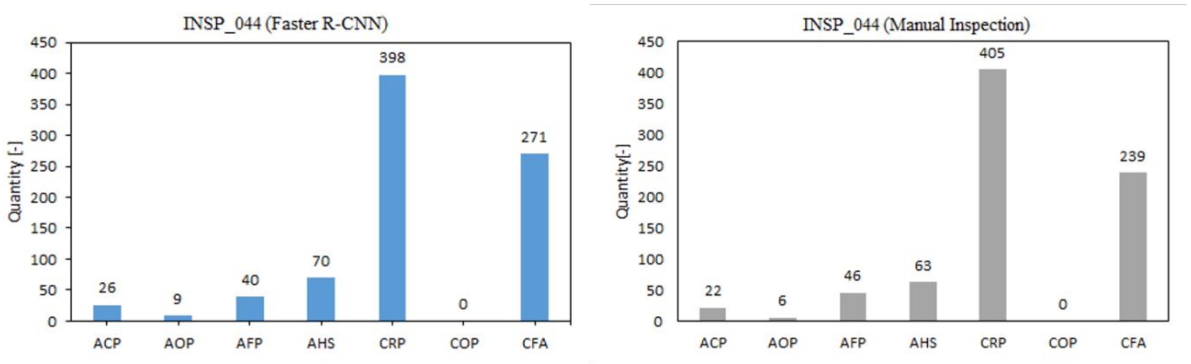
Data availability is another critical factor to consider when selecting a model for any machine learning problem. More importantly, in the case of defect detection, where it can become very time-consuming and expensive to obtain sufficient training data, is the data efficiency of the learning algorithm. For simple tasks like differentiating cats from dogs, humans take very few examples to differentiate the two. However, CNNs typically require at least a few hundred examples to get reasonable performance and thousands of examples to get to human-level performance. A simple explanation for this is that the network cannot focus on what we know to be relevant, and they only generalize by finding common patterns in the given examples. For instance, if all the images of dogs shown to the network were taken on green grass, the network will learn to associate the green colour with the dog class. Only when the network is presented with examples of dogs from other surroundings will the network start to ignore those irrelevant features. Therefore, in general, the more difficult the underlying pattern is to recognise, the more diverse examples the networks typically need to be shown. In our case, we took a top-down approach where we first trained a simple classification network to differentiate images with defects from images without defects. This allowed two things: 1) it allowed us to develop a baseline model to compare the performance of the deep learning model to classic image process approaches. 2) The trained network was used to collect more training examples for retraining and improving the model. By doing this, we were able to increase the complexity of the task and the complexity of the network since we now have a larger training dataset. In Figure 4-13, we showed how the size of the training dataset affects the mAP on the testing set for the Faster R-CNN and SSD model. In this case, the performance of the two-stage model was much better (Faster R-CNN mAP 0.78, SSD mAP 0.52) for smaller datasets (less than 1600 images), while the single-stage network only started to catch up as the size of the dataset increased (greater than 2400 images). However, compared with the image classification models, it is much more time-consuming to label tight-fitting bounding box annotations for training object detection models.

Runtime evaluation

For most modern object detection systems, there is an inherent trade-off between speed and accuracy [168], and we observed a similar behaviour as shown in Figure 4-14, where the number of region proposals generated by the RPN affects the speed and accuracy of the Faster R-CNN network. Nevertheless, it was possible to reduce the number of proposals from 6000 to 3000 without any drop in mAP. Consequently, this reduced the time taken to evaluate a single image of size 800 x 600 pixels from 438 ms to 341 ms. However, this is still much higher than the 121 ms it takes the SSD model to process a single image, making the Faster R-CNN network ~3 times slower. It takes the Faster R-CNN based system ~35 minutes and the SSD based system ~12 minutes to process an entire cross-section using a GPU. On the other hand, the sliding window classifier takes almost 3 hours to process a full cross-section image.

5.1.2 Comparison between manual inspection and the proposed method

In our previous experiments, we evaluated the performance of the defect detection system based on a set of ground truth annotations provided by a human inspector. However, this only checks how well the system performs on annotated examples; it does not give any indications of missed defects since they were not annotated. Therefore, this section will compare the performance of the model to manual inspection on two cross-sections, which will give a better indication of the general performance of the model and the time taken for the analysis. Figure 5-2 shows a side-by-side comparison of the Faster R-CNN with the ResNet-50 based defect detection system to the results of manual inspection for each defect category. In general, both results are highly comparable; however, on average, it takes ~1.5 hours to manually inspect each cross-section image compared to ~35 minutes taken by the defect detection system. Moreover, we also observe the speed-accuracy trade-off here, where the more time spent manually analysing the images, the more accurate the result is. However, this level of concentration cannot be sustained for too long without the possibility of fatigue-induced error, which is not the case for our defect detection system.



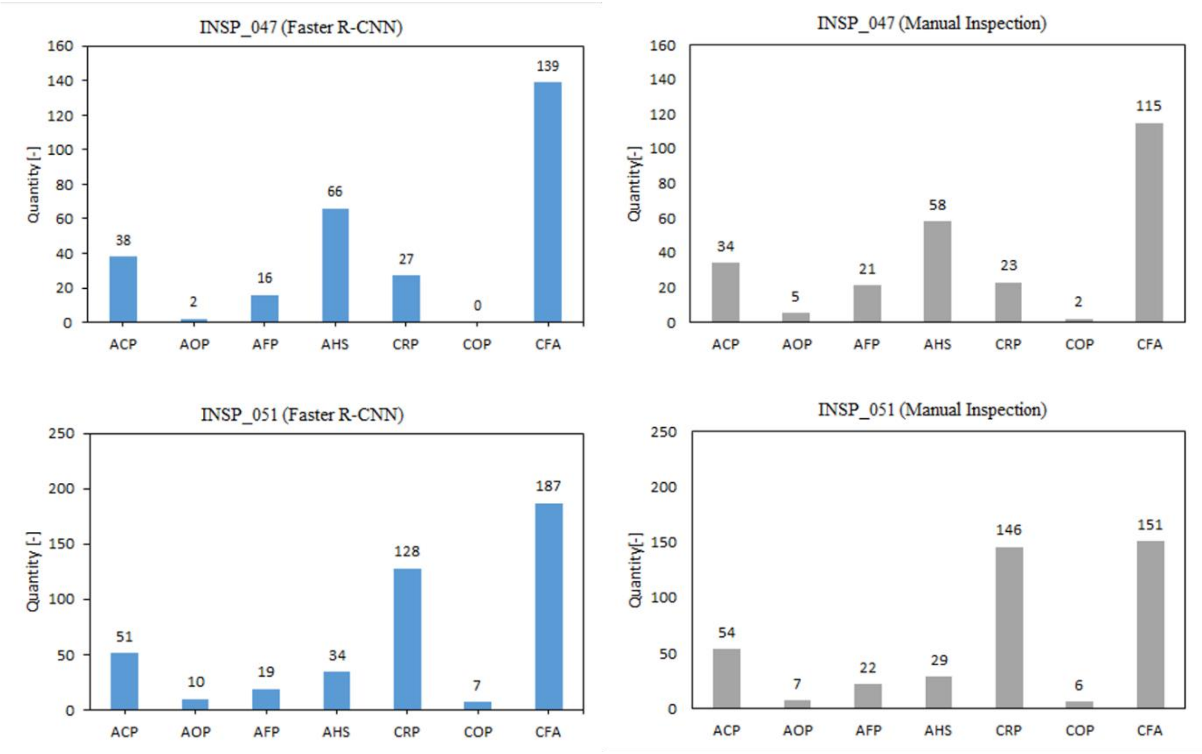


Figure 5-2: Compares the number of defects detected by the proposed system and manual inspection for the three battery samples INSP_044, INSP_047 and INSP_051 (two cross-sections each) presented in Table 4-6.

5.1.3 Comparison between the proposed methods and classical approaches

In [57], we compared the performance of various CNN models built from scratch with randomly initialised weights to state-of-the-art models initialised with pretrained weights and traditional machine learning approaches using general texture extraction methods with shallow classifiers. Concretely, first, a baseline model was developed as a reference to the traditional approaches and based on the results, two optimised models were developed, one with a sigmoid layer at the output and the other with a softmax layer at the output. For the traditional approach, two models were evaluated; the first method uses a Gray-level co-occurrence matrix (GLCM) as the feature extractor and a support vector machine classifier, while the second model was based on a multiresolution local binary pattern (MLBP) with a gradient boosting classifier [180]. In addition, we also evaluated the performance of state-of-the-art models, which were pre-trained on millions of images from the ImageNet dataset and fine-tuned for our defect detection task. The results are summarised in Table 5-1. In general, the fine-tuned models pre-trained on millions of images achieved a much better performance than the CNN models trained from scratch with randomly initialized weights, with the VGG-19 model achieving the best performance overall (F_1-score 0.99 for defect class and 1.0 for no defect class). On the other hand, the traditional method based on GLCM with SVM classifier achieved the lowest performance (F_1-score 0.46 for defect class and 0.91 for no defect class), while the model based on MLBP with gradient boosting classifier achieved a comparable result to the baseline model but performed significantly worse compared to the other CNN models. These results demonstrate the major benefits of transfer learning, where low-level features learnt from one dataset can be transferred to another dataset, especially when only a few examples are available in the downstream task. In contrast, the CNN models trained from scratch directly on the battery data could not learn sufficiently general features due to the small dataset and therefore are more likely to overfit the training data. On the other hand, the features extracted by the traditional approaches are generally fixed and aim at a specific pattern, lacking adaptability and robustness for such classification tasks. However, due to the large model size, the inference speed is much slower for the fine-tuned models.

Table 5-1: Classification results for the Li-ion battery micrographs dataset. Class 0 represents examples with defects, and Class 1 represents examples with no defects. Fine-tuned VGG-19 model performed best, followed by the InceptionV3 [57].

Model	F1-score		AUPRC		AUROC
	Class 0	Class 1	Class 0	Class 1	
GLCM + SVM	0.46	0.91	0.372	0.859	0.653
MLBP + Gradient Boosting	0.73	0.95	0.614	0.918	0.810
Baseline	0.71	0.91	0.815	0.977	0.930
Sigmoid	0.86	0.97	0.957	0.996	0.980
Softmax	0.85	0.96	0.924	0.992	0.970
VGG-19 fine-tuned	**0.99**	**1.00**	**0.999**	**1.000**	**1.000**
InceptionV3 fine-tuned	0.97	0.99	0.993	0.999	1.000
Xception fine-tuned	0.95	0.99	0.960	0.989	0.980

5.1.4 Benefits of the proposed method for quality assessment of batteries

Besides the ability to quantitatively and qualitatively analyse a battery sample for the presence of defects and compare samples from various producers for quality assessment, another major benefit of the proposed approach is that we also obtain the exact coordinates of the detected defects. These can later be used to verify the chemical composition of the detected region using correlative analysis techniques. Moreover, for battery producers and researchers, there is little benefit in simply being able to visually observe the presence of something without the ability to eventually determine its origin since further information would be needed to take corrective measures. Therefore, in this section, a simple workflow is presented to demonstrate the benefits of the proposed detection system.

As described in Section 3.10, our defect localisation network produces three outputs for each detected defect: the class of the defect, the confidence score and the bounding box coordinates. Subsequently, we can use the bounding box coordinates to crop the whole defect region and automatically sort the defects into various categories using the predicted class labels to obtain a picture gallery such as those shown in Section 4.2.4 and produce a statistical report. In the next step of the workflow, our image selection tool is used to select a subset of defects from the picture gallery in order to mark the regions corresponding to the selected defects on the full cross-section, as shown in Figure 5-3. This is much faster than trying to locate each of those areas manually. Finally, with the marked image of the battery cross-section and the corresponding prepared battery sample, we can perform correlatively analysis with the scanning electron microscope by using the markers on the light microscopy image to locate the equivalent points on the battery sample for analysis. Figure 5-4 shows the results of using EDX analysis to determine the chemical composition of a selected number of detected defects. For example, in Figure 5-4 (a), the detected defect at point A1-P1 contains a higher oxygen content (7.0%) than the graphite particles at points A1-P4 (3.9%) and A1-P5 (4.2%). Likewise, in Figure 5-4 (b), the detected defect at point A1-P1 contains a higher oxygen (25.0%) and fluorine (5.9%) content but a lower carbon (64.6%) content than the graphite particle at point A1-P6 (4.2%, 0.3% and 91.9%, respectively). However, the main component is carbon which is why defects of this kind were named anode carbon particles (ACP). In Figure 5-4 (c), the detected particle contains a high content of silicon (17.0%) and zirconium (14.4%), while the highest component is oxygen (67.3%), hence the name anode oxide particles (AOP). Following this approach, we were not only able to identify and properly categorize the detected defects, but we were also able to detect possible cross-contamination of different NMC compositions. For example, in Figure 5-4 (d), the results of the EDX analysis of the large round particles in the cathode show that the coating material of the cell has a chemical composition consisting of NMC-622 at point A1-P3 (Ni 46.6%, Mn 16.2%, Co 16.6%,) while the large round particle contains a different chemical composition consisting of NMC-424 at point A1-P1 (Ni 28.6%, Mn 15.0%, Co 24.2%,). Furthermore, we observed that the number of these large round particles also varies significantly depending on the production date, as shown in Figure 5-5, which would indicate possible cross-contamination from a different production batch.

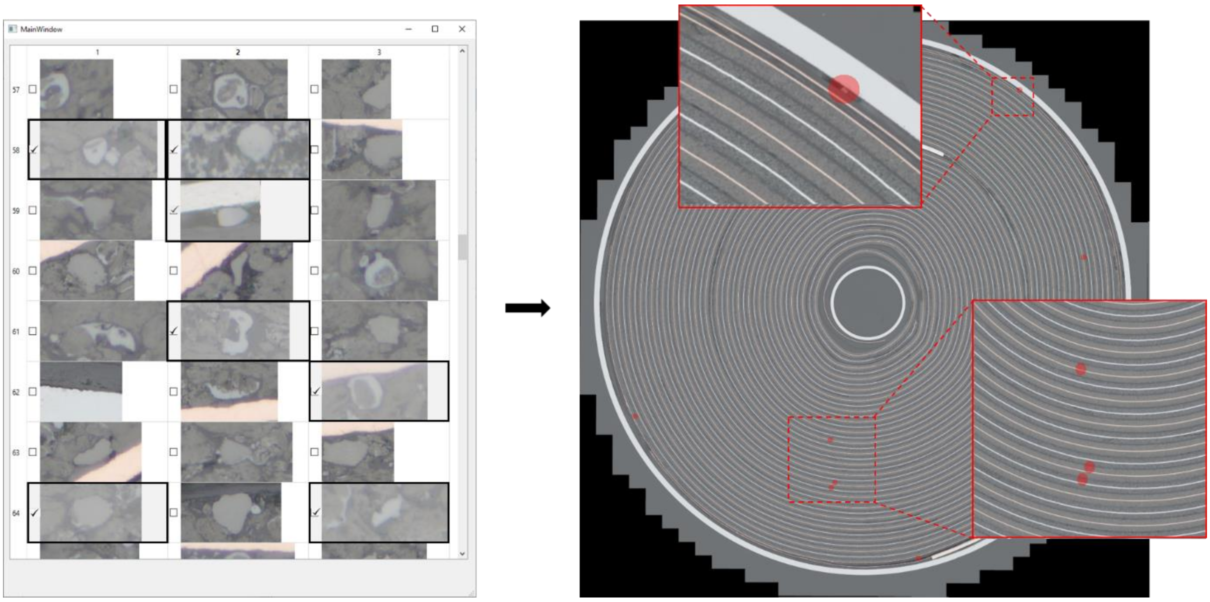
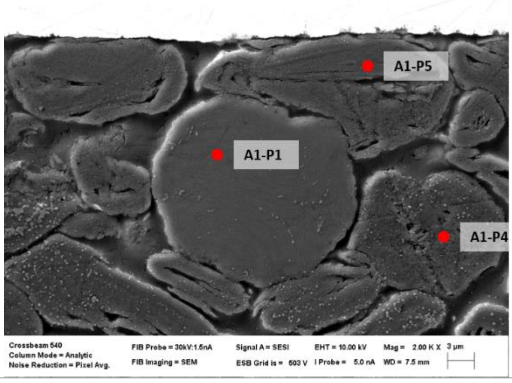
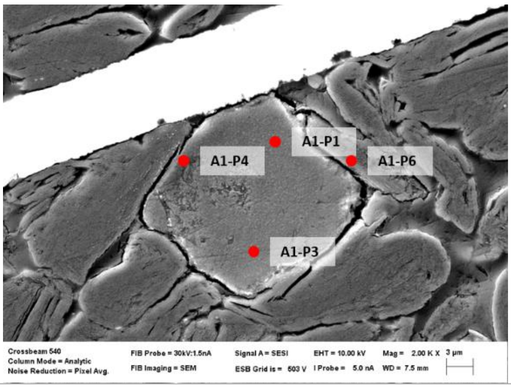
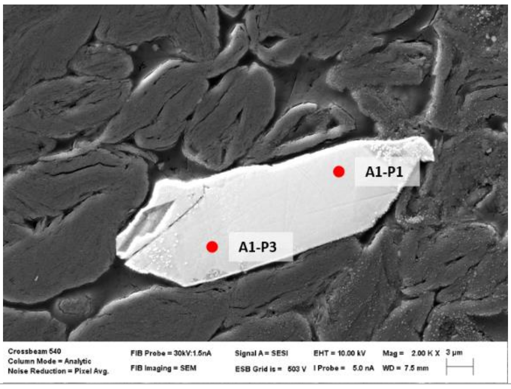
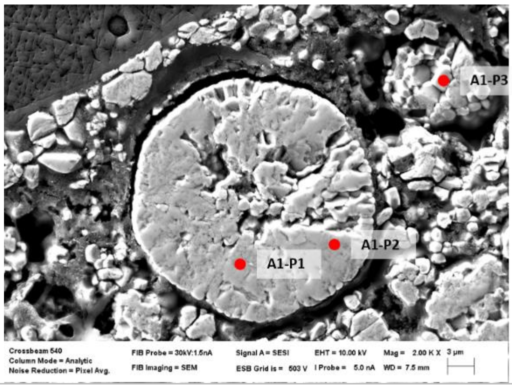
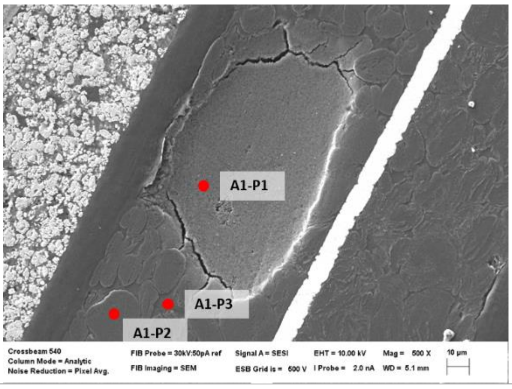
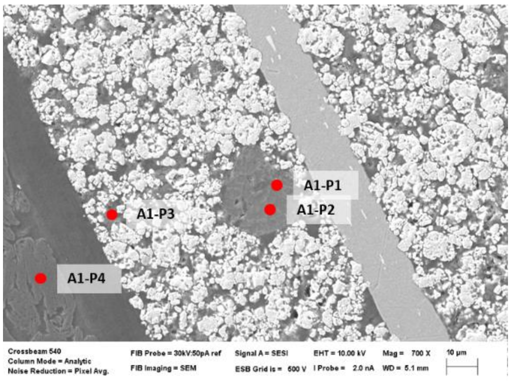


Figure 5-3: This depiction shows the image selection process for correlative analysis. Our image selection tool (left) can be used to select a subset of defects from the picture gallery containing all the detected defects. The coordinates of those selected defects are subsequently marked in red circles on the large overview image (right).

(a) 	 A1-P5 A1-P1 A1-P4 Crossbeam 540 Column Mode = Analytic Noise Reduction = Pixel Avg. FIB Probe = 30kV:1.5nA FIB Imaging = SEM Signal A = SESI ESB Grid is = 603 V EHT = 10.00 kV I Probe = 5.0 nA Mag = 2.00 K X WD = 7.5 mm 3 µm	Phase fractions at. % A1-P1 C 92.6, O 7.0, S 0.2, Au 0.2 A1-P4 C 95.3, O 3.9, F 0.6, Au 0.2
(b) 	 A1-P1 A1-P4 A1-P6 A1-P3 Crossbeam 540 Column Mode = Analytic Noise Reduction = Pixel Avg. FIB Probe = 30kV:1.5nA FIB Imaging = SEM Signal A = SESI ESB Grid is = 603 V EHT = 10.00 kV I Probe = 5.0 nA Mag = 2.00 K X WD = 7.5 mm 3 µm	Phase fractions at. % A1-P1 C 64.6, O 25.0, F 5.9, Ni 2.4 A1-P6 C 91.9, O 4.2, F 0.3
(c) 	 A1-P1 A1-P3 Crossbeam 540 Column Mode = Analytic Noise Reduction = Pixel Avg. FIB Probe = 30kV:1.5nA FIB Imaging = SEM Signal A = SESI ESB Grid is = 603 V EHT = 10.00 kV I Probe = 5.0 nA Mag = 2.00 K X WD = 7.5 mm 3 µm	Phase fractions at. % A1-P1 C 0.5, O 67.3, Si 17.0, Zr 14.4
(d) 	 A1-P3 A1-P2 A1-P1 Crossbeam 540 Column Mode = Analytic Noise Reduction = Pixel Avg. FIB Probe = 30kV:1.5nA FIB Imaging = SEM Signal A = SESI ESB Grid is = 603 V EHT = 10.00 kV I Probe = 5.0 nA Mag = 2.00 K X WD = 7.5 mm 3 µm	Phase fractions at. % A1-P1 O 31.1, Mn 15.0, Co 24.2, Ni 28.6 A1-P3 O 19.4, Mn 16.2, Co 16.6, Ni 46.6

<table>
<tr>
<td>(e)</td>
<td rowspan="2">


</td>
<td rowspan="2">Phase fractions at. %
A1-P1 C 86.0, O 2.8, F 1.7, S 2.0, Cu 6.3
A1-P2 C 98.7, O 0.1, F 0.2, Cu 0.9
A1-P3 C 68.5, O 11.3, F 15.4, Cu 2.5</td>
</tr>
<tr>
<td> </td>
</tr>
<tr>
<td>(f)</td>
<td rowspan="2">


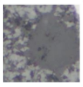
</td>
<td rowspan="2">Phase fractions at. %
A1-P1 C 3.5, O 35.3, F 55.0, Cu 4.7, Al 0.7
A1-P3 C 75.3, O 16.9, F 7.1, P 0.5, Au 0.2</td>
</tr>
<tr>
<td></td>
</tr>
</table>

Figure 5-4: Correlative analysis using light microscopy (left) and SEM (middle) to determine the chemical composition of detected defects. The result of the EDX analysis is shown on the right. Point A1-P1 corresponds to the defect detected by the model, and the other points are for comparison.

Regarding robustness, an important question is how well the detection system works when it receives images from batteries with different chemistries or microstructures than that of the model training data set? Figure 5-7 shows the microstructures of four battery samples with different chemistries from the one with which our system was originally trained. The sample with the most similar microstructure to the LG Chem ICR 18650 B4 NMC is the Samsung INR 18650 30Q NCA, which we use to test the generalizability of our detection system. For the other samples, such as the Sony US 18650 VTC5A NCA with silicon/carbon composite in the anode or the Sanyo UR 18650 W2 NMC/MnO with blend cathode to enhance the energy density and thermal stability, the microstructure of these batteries look significantly different from those of the LG Chem ICR and when presented to the detection system results in a lot of false positive detection. This is unsurprising since large particles in the cathode or foreign particles in the anode are often labelled as defects in our training data. As a result, in such cases, we cannot directly apply the system without retraining with some new examples to adjust the models' weights to adapt to the new microstructure. Nevertheless, we can leverage transfer learning for a much quicker training process. On the other hand, the system worked directly without retraining on the Samsung INR18650-30Q NCA samples, which have different chemistry to our original training data. The distribution of the detected defect from two cross-sections of this sample is shown in Figure 5-6. We can immediately observe significant differences in these results from those of the LG Chem ICR in Section 4.2.2. For instance, the number of CRP and CFA defects is much lower in comparison, while the quantity of ACP defects is much higher than in the LG Chem ICR samples.

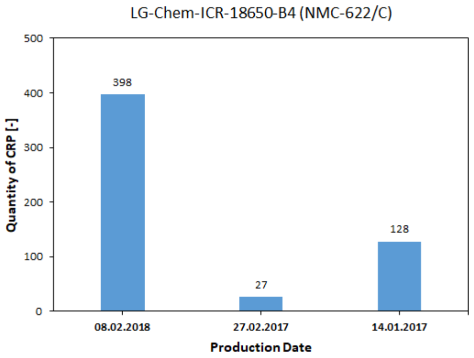


Figure 5-5: shows the quantity of CRP defects detected in samples INSP_044, INSP_047 and INSP_051 with production dates 08.02.2018, 27.02.2017 and 14.01.2017, respectively.

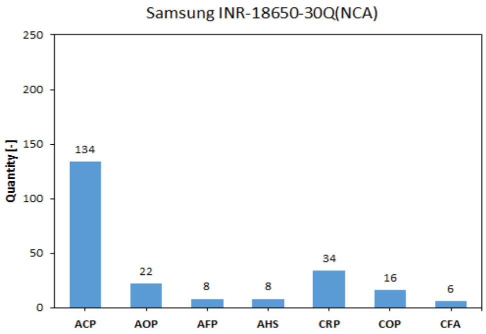


Figure 5-6: Shows the distribution of defects detected in 2 cross-sections of the Samsung INR-18650-30Q (NCA) sample.

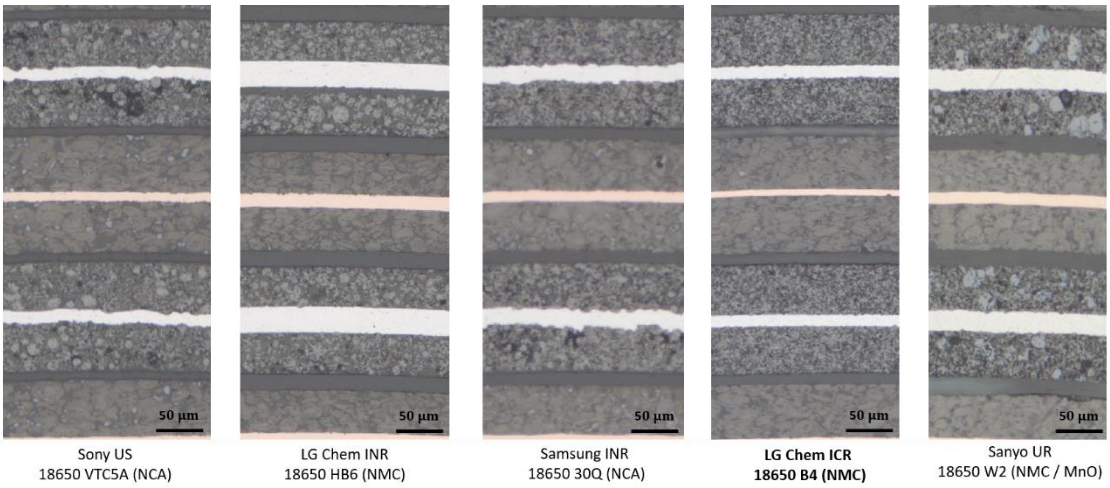


Figure 5-7: Example images of microstructure from batteries with different chemistry.

5.1.5 Limitations of the proposed method for quality assessment of batteries

This section will discuss some of the limitations associated with the proposed defect detection system. First and foremost, since the image acquisition process is destructive, the packaged cell needs to be sectioned (cut open) to reveal the electrodes; therefore, it is impossible to carry out any further investigation to determine the effects of these defects on the performance of the battery. For example, we cannot perform cycling tests to find specific correlations between the defects and the lifetime of the cell. Secondly, the sample preparation and image acquisition process can be time-consuming. Nevertheless, we envision a use case where a small batch of samples is randomly picked from a large production batch for analysis or a few samples from various producers are periodically compared for quality assessment. However, with a sufficiently high-resolution camera, such a system can be deployed during production since the detection model can operate in near real-time. In addition, due to the delicate composition of materials within the battery, preparing cross-sections free of artefacts can be very challenging. However, the model can be trained to identify certain artefacts and ignore them, while the origin of other types of defects such as cracks and geometric irregularities can be more difficult to determine since they might be actual defects or introduced during the sample preparation process. Lastly, as discussed in the previous section, in order to apply the system on a battery with different chemistry or microstructure from the one with which the model was originally trained, the model will need to be retrained or fine-tuned with examples of the new battery microstructure. Although our training dataset consists of examples from only one type of battery chemistry, with a sufficiently diverse training set, the system can probably be applied to a broader range of microstructures without the need for retraining.

5.2 Detecting defects in sintered NdFeB magnets

In the previous sections, we looked at various techniques for automated defect detection using a supervised learning approach and discussed several advantages of using this method. However, one of the main limitations of using this approach is the need for a relatively large amount of labelled data, and depending on the type of supervised learning approach, this can range from simple class labels to time-consuming bounding box annotations. Furthermore, in certain situations where one class is abundant and very few examples of the other class exist, it is typically not feasible to train a supervised learning model or in cases where prior knowledge of what constitutes acceptable or defect pattern is not well consolidated. For these reasons, two unsupervised learning approaches were investigated to learn the structure of normal images from the vast quantity of unlabelled data to detect irregularities in the microstructure of sintered NdFeB magnets. In this section, the performance of these two models will be evaluated and compared to manual inspection and other anomaly detection techniques. Then the relevance and benefits of the proposed systems will be discussed, and in the final section, some of the limitations will be presented.

5.2.1 Performance assessment of proposed methods

Variational autoencoders (VAEs) are designed to learn both an encoder and decoder network, which often results in a good ability to reconstruct the training data as well as the ability to quantify a bound on the log-likelihood fit of the model to data [130], [131], [181]. Moreover, the inferred latent codes can be utilized in downstream tasks in a semi-supervised learning approach for image classification [182]. However, new images synthesized by the VAE tend to be rather unspecific and blurry with relatively low resolution, as we saw in Section 4.3.1. This is because the VAE seeks to maximize a lower bound on the log-likelihood of the generative model, which means they inherit the limitations of maximum-likelihood learning [183]. In particular, for training the VAE, we optimize the Kullback-Leibler (KL) divergence between the underlying data distribution and the distribution of the model. In such a setting, the learning approach does not penalize the model for generating outputs that are different from those used for training, which is compensated for by adding the mean squared error to the final loss. Nevertheless, compared to a normal autoencoder where latent variables' distribution for different classes are not very well distributed and with continued training, the autoencoder inevitably reduces the reconstruction error of outliers. This allows the autoencoder to easily reconstruct any input image from the training dataset, which is not particularly useful for detecting anomalies in images due to data leakage [184], [185]. On the other hand, the VAE can learn a better distribution of the underlying data where the distribution of latent variables for different classes are evenly spread and overlap slightly with each other to create a continuous transition. This ensures to some extent that images not drawn from the same distribution as the training data will typically have a much higher reconstruction error. However, for our particular application, which involves analysing thousands of tile images, the fact that VAEs tend to generate unspecific and blurry images results in an undesirable amount of false positives, where images that do not contain any defects sometimes have a high reconstruction error due to randomness in the data generation process.

On the other hand, GANs are the state-of-the-art approach for generating photorealistic images, and we leverage this approach to improve our defect detection models based on the practical limitations of the VAE model. In particular, due to the constraint imposed on the latent vector z, which forces the network to learn a compressed representation of the input data, resulting in blurry and sometimes random outputs, we use a conditional GAN to learn a direct mapping from edges to images. Furthermore, the U-Net generator, which uses skip connections to shuffle low-level features shared between the input and output across the network, allows the model to generate high-resolution outputs. Another major benefit of this approach is that the model produces deterministic outputs for a given set of input images. Although various image-to-image translation approaches with conditional GANs try to mitigate this by either providing Gaussian noise as an input to the generator or including several dropout layers in the generator during training and testing, they observed only minor stochasticity in the output of the network [160], [164], [186]. However, for anomaly detection, it is highly beneficial for all parts of the original and generated image to be identical except for the anomalous regions, which helps reduce the amount of false positive detections significantly.

Next, in order to investigate how well our model generalises to other types of magnet microstructures and imaging conditions, our defect detection model based on the conditional GAN was applied to two new NdFeB magnet samples, HS17051_75 and HS17051_03, which have a slightly different microstructure and contrast to those initially used for training the model. A comparison of the microstructures of the new testing samples with the original ones used for training is shown in Figure 5-8. The model was directly applied to the two samples without retraining with example images from the new dataset to determine how well the model performs on new microstructures. Figure 5-9 shows some examples of the detected defects from both samples. For HS17051_03, out of 35489 patches analysed, the system reported 169 patches containing defects, of which 125 were true positives, 29 were false positives, and 15 were preparation artefacts. For the HS17051_75, out of 20321 patches analysed, the system reported 396 patches containing defects, of which 349 were true positives, 43 were false positives, and 3 were preparation artefacts. The results indicate that the model can generalise to new magnet microstructures without any retraining; however, we observed a drop in the model performance when there is a significant difference in image contrast. Nevertheless, this can be solved by either fine-tuning the model with some examples from the new dataset or retraining the original model using grayscale images.

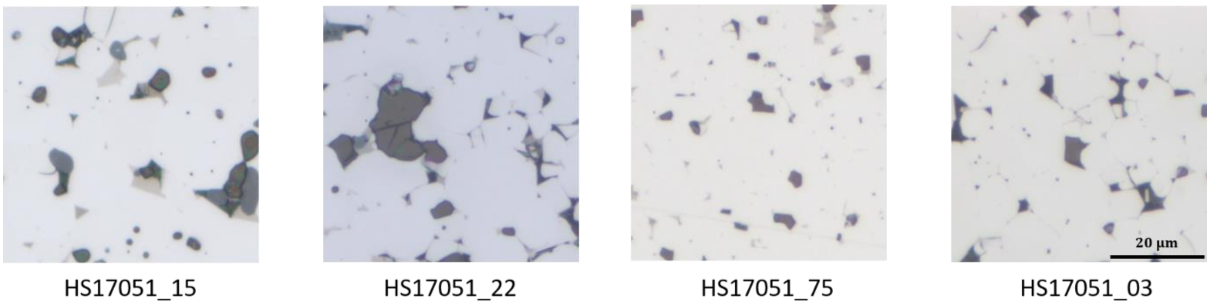


Figure 5-8: Compares the microstructure of the samples used for training the model to two new samples, HS17051_75 and HS17051_03, with different microstructures for testing.

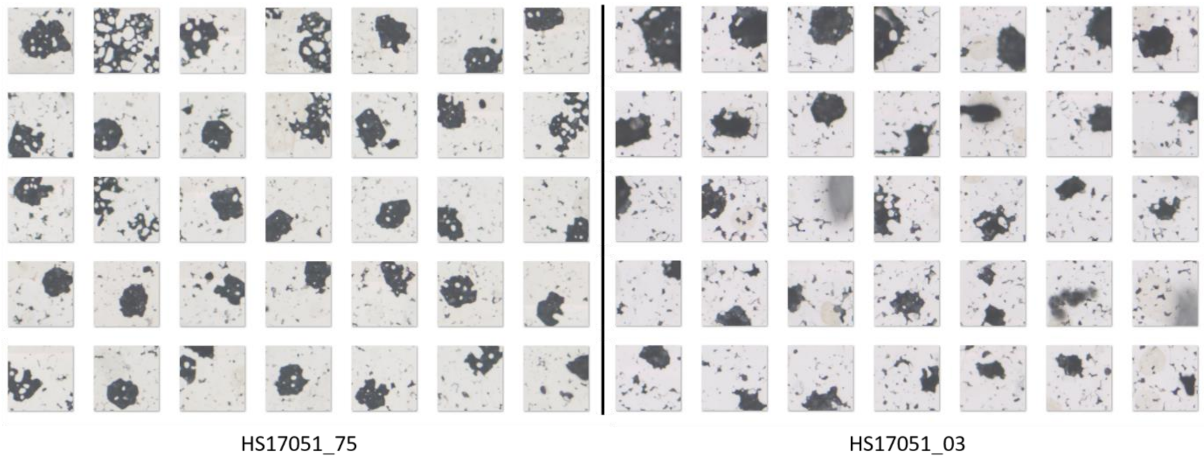


Figure 5-9: Shows some examples of defects detected by the cGAN model (without retraining) in the magnet samples HS17051_75 and HS17051_03, which have a different microstructure from those originally used for training the model.

Runtime evaluation

Both defect detection approaches take nearly the same amount of time to analyse a full cross-section image. For example, to analyse the HS17051_03 sample, which contains 35489 patches, it took the VAE based system ~2.35 hours while the system based on conditional GAN took ~2.5 hours. The time it takes to analyse an image heavily depends on the size of the window strides, where a stride of 1 indicates no overlap between adjacent patches and a stride of 0.5 indicates a 50% overlap between adjacent patches. Using a certain amount of overlap helps to handle border effects and improves model performance. Thus, we used an overlap of 75% in all our analyses to balance speed and detection accuracy.

5.2.2 Comparison between manual inspection and the proposed method

Next, the performance of the conditional GAN model is compared to manual visual inspection to evaluate how well the model performs compared to a human examiner. For this, the samples HS17051_90 and HS17051_35 were visually inspected, and the time taken for each analysis was recorded. The results of the analyses are shown in Figure 5-10. For the sample HS17051_90, a similar number of true positives were detected through manual inspection and our unsupervised model. The difference in the number of reported defects is mainly because some defects are split across multiple windows and hence counted separately by our detection system, while those are only counted once during the manual inspection (see Figure 4-32 and Figure 4-33). Note that preparation artefacts are counted during manual inspection only to compare the result with the automated system. However, for the HS17051_35 sample, more defects were detected through manual inspection than our detection system reported. Nevertheless, as shown in Section 4.4.4, a considerable amount of the detected defects were at the border of the sliding window, which means that only a smaller part of the defect is considered in the reconstruction error for that window. This indicates that we can increase the model's accuracy by simply using a smaller stride to increase the amount of overlap between adjacent windows so that defects are more likely to fall in the centre of a window. Therefore, the window stride was reduced from 0.75 to 0.5, and the result is shown in Figure 5-11. Here it can be seen that the model’s performance improves, although at the cost of a much longer processing time due to higher overlap between adjacent tiles.

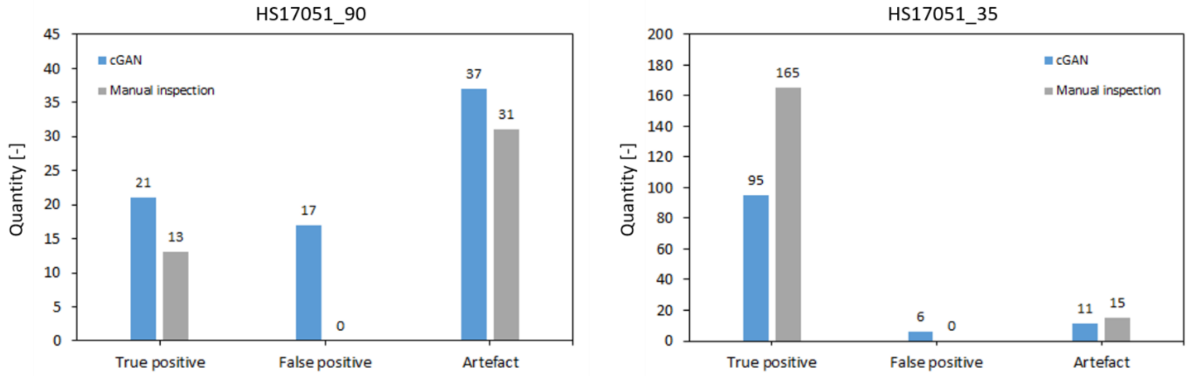


Figure 5-10: Compares the number of defects detected by the proposed system to manual visual inspection for the samples HS17051_90 and HS17051_35. Note that the number of preparation

artefacts (Artefact) was counted during manual inspection simply to compare the result with the trained model.

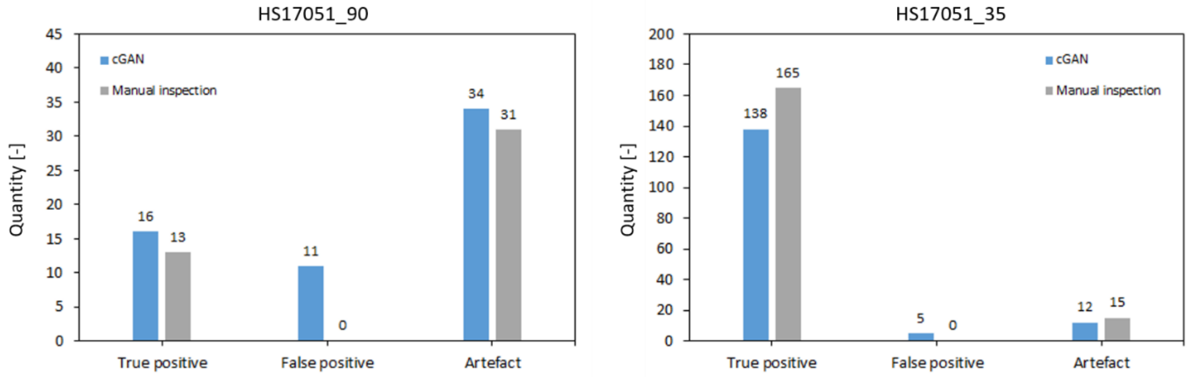


Figure 5-11: Shows the number of defects detected in the samples HS17051_90 and HS17051_35 by the proposed system compared to manual visual inspection after adjusting the window stride from 0.75 to 0.5.

5.2.3 Comparison between the proposed methods and classical approaches

Many of the methods available for identifying anomalies focus mainly on small datasets with a low number of features. One-Class Support Vector Machines (OC-SVM) [187]–[189], Isolation Forest [190], and Local Outlier Factor [191] are popular techniques for unsupervised anomaly detection. Of the three techniques, OC-SVM is the most common approach for anomaly detection. These techniques generally aim to model the underlying distribution of normal data while being insensitive to noise or anomalies in the training data. For instance, in the case of OC-SVMs, a kernel function is used to implicitly map the input space to a higher dimensional feature space in order to clearly separate normal and anomalous data. Theoretically, SVMs are appealing because they provide good generalisation when the parameters are properly configured. Since the loss function is convex, they deliver a unique solution, and they can model any training set in principle when an appropriate kernel is chosen [192], [193]. However, these methods often fail in high-dimensional, data-rich settings due to bad computational scalability and the curse of dimensionality [97], [194]. The curse of dimensionality implies that for a model to obtain good generalization, the number of training samples must grow exponentially with the number of features [195], [196].

However, SVMs are non-parametric models whose complexity scales quadratically with the number of records, making the training both memory and time-intensive. Hence they are best suited to small datasets with few features since a large number of features, for example, in image data, results in the curse of dimensionality, which causes the generalization error of shallow architectures such as SVMs, to increase with the number of irrelevant and redundant features. In addition, shallow architectures have practical limitations for modelling certain types of input functions. In order to avoid these issues, it is necessary to generate a compact representation of the underlying data distribution, which can capture most of the variations in the input data to alleviate the curse of dimensionality and reduce the computational complexity of the algorithm. A common approach to address this problem is through a hybrid approach where dimensionality reduction techniques such as principal component analysis (PCA) [197], independent component analysis (ICA) [198] or Deep Belief Nets (DBNs) [132], [199], [200] is used to extract features for training a One-Class SVM [97], [201]. Another common approach used for defect detection in textured surfaces involves the extraction of second-order statistical attributes (Haralick features) derived from grey-level co-occurrence matrices (GLCM) to train a One-Class SVM to learn a general description of normal texture from defect-free samples [202]. However, such shallow methods typically require substantial feature engineering to be effective. Moreover, as we have seen previously, techniques that rely on traditional hand-crafted features tend to have much lower accuracies when compared to deep learning based methods.

On the other hand, deep hybrid models for anomaly detection mainly use autoencoders for feature extraction, whereby the features learnt within the hidden representations of autoencoders are fed as input to anomaly detection algorithms such as OC-SVM to detect outliers [203]–[205]. The feature extractor greatly reduces the curse of dimensionality, which makes hybrid models more scalable and computationally efficient since the linear or nonlinear kernel models operate on reduced input dimensions. However, a hybrid approach is unable to influence representations learnt within the hidden layers of the feature extractors since generic loss functions are typically employed instead of customized objectives for anomaly detection; hence these models often fail to extract differential features to detect outliers.

5.2.4 Benefits of the proposed method for quality assessment of magnets

We proposed a generic unsupervised learning approach capable of anomaly or defect detection in image data. The proposed method can handle high-dimensional data without making any prior assumptions on the underlying data distribution, besides ensuring that the training dataset consists of only defect-free examples. In contrast, most unsupervised anomaly detection models require priors to be assumed on the distribution of anomaly, which typically makes the models less robust to handle noisy data. Compared to conventional autoencoder-based anomaly detection approaches, which are highly sensitive to even the slightest violations of the clean-dataset assumptions, if a small number of anomalies contaminate the training set, our approach will not necessarily learn to reconstruct anomalous observations as well as normal ones. In addition, our approach is not unduly affected by small deviations from the training distribution. That is, it is robust enough to deal with microstructures from different samples of magnets without requiring further retraining. Unlike the supervised learning models we discussed in the previous sections, our unsupervised models can discover unknown defect patterns in the data, making it very useful, especially in cases where the knowledge of what constitutes a defective sample is not well established. Furthermore, supervised deep learning techniques often require many training samples with accurate labels to effectively learn discriminative features for various classes. However, accurate labels for various normal and anomalous instances are often not readily available for many practical applications, particularly for defect detection. In many cases, they are costly and time-consuming to acquire, which can make training a robust supervised model a very challenging task. Nevertheless, as shown in Section 4.4, by using our proposed method, we can obtain a picture gallery of all the detected defects, which provides a snapshot of the state of the magnet and therefore facilitates qualitative and quantitative assessments of magnets of different grades from various producers which is highly beneficial for many industrial applications. For example, original equipment manufacturers can make more informed decisions based on the types and quantity of defects in the analysed magnet samples; since a high population of defects will normally correlate with a worse manufacturing process and hence a lower quality magnet that easily loses its magnetic properties.

5.2.5 Limitations of the proposed method for quality assessment of magnets

High-dimensional problem domains still pose significant challenges for anomaly detection. This is mainly due to the presence of many irrelevant features which conceal the presence of anomalies and the absence of a clear objective function for learning anomaly detection, which still typically relies on reconstruction error based heuristics. Nevertheless, we have been able to successfully mitigate this problem by developing a model that can reconstruct high-resolution versions of the input data and using a good similarity measure that is less sensitive to slight deviation in pixel values. However, there are still certain scenarios in which our model is suboptimal. For instance, with our approach, detecting low contrast defects such as the presence of secondary phase and regions with large grains will be challenging since they appear very similar to the main hard magnetic phase $Nd_2Fe_{14}B$. A possible solution would be to use other imaging techniques which normally have higher contrast, such as Kerr microscopy or by staining the sample with compounds that give these regions higher contrasts. Nonetheless, it is also possible to train a supervised model to detect the secondary phase since these defects are often of much higher quantity than the large grains or oxides. Another limitation of the proposed approach is the time taken to evaluate a sample (~2.5 hours), which can easily become a bottleneck if lots of samples need to be analysed. However, using various optimisation techniques and parallel processing can reduce the evaluation time significantly.

Chapter 6

Conclusion

Throughout this thesis, we have investigated various methods for automating the detection of microstructural defects in two types of energy-related technologies, namely: lithium-ion batteries and sintered NdFeB magnets. In this final chapter, we will review the initial problem and recapitulate the proposed methods (Section 6.1), summarize our findings (Section 6.2), discuss the impact of machine learning on microscopy (Section 6.3) and provide an outline of future research directions (Section 6.4).

6.1 Synopsis

Quality control focuses on identifying defects in products and monitoring activities to verify that products meet the desired quality standard. However, quality inspection of components is often labour intensive and can be error-prone due to the limits of human visual perception. As a result, automated vision systems are becoming essential in many production facilities. For this task, it is often common to develop solutions predominantly based on manually engineered features that rely on expert knowledge to handcraft useful features. However, this process can be laborious and costly while the produced solution is still brittle, needing significant modifications for slightly different use cases. Moreover, quality evaluation techniques for assessing the quality of electrode foils of lithium-ion cells currently applied in the industry do not provide sufficient detail or spatial resolution on microstructural features that are critical for battery performance. To this end, we proposed a more general data-driven approach based on recent advances in computer vision technologies using convolutional neural networks to learn representative features directly from the given data.

In chapter 3, we developed models for defect detection, which can be broadly categorized into supervised and unsupervised deep learning techniques. In Section 3.8, a simple binary image classifier based on a sliding window approach was developed to detect defects in the microstructure of Li-ion batteries. This approach was later extended to handle multiple classes of defects in Section 3.9. Our final model enabled the detection of defects with high accuracy and allowed us to visually inspect the model's predictions and investigate possible sources of ambiguities in the data. However, due to certain limitations and, more importantly, our findings from these initial models, in Section 3.10, more complex object detection and localisation models were developed based on one-stage and two-stage detectors that can detect and localize multiple classes of defects in the same image with high accuracies in near real-time. All the above techniques fall under the supervised learning approach, which primarily relies on having a relatively large labelled training dataset. However, in many cases, such datasets are not readily available. Therefore, in Section 3.11.1, an unsupervised model was developed for anomaly detection in the microstructure of sintered NdFeB magnets without the need for any labelled training data. The model can detect defects by learning from the training data the underlying distribution for “normal” microstructure patterns in order to reconstruct the original image. Therefore, when the model receives an image with an unusual pattern (defect), it generates a high reconstruction error. Finally, in Section 3.12, the anomaly detection accuracy was significantly improved by developing a system that can better reconstruct the original input image in high resolution by combining two networks in an adversarial framework.

6.2 Summary of findings

This thesis aimed to develop new approaches for performing the quality evaluation of Li-ion battery electrodes and sintered NdFeB magnets by detecting various microstructural defects from light optical microscopy images. Various deep learning model architectures were investigated for detecting defects in Li-ion batteries, and with our proposed solution, we successfully detected various categories of defects with high accuracy. For the binary classifier model, which categorizes images into a defect or no defect class, the ResNet-50 model trained with the focal loss objective using hyperparameter $\alpha = 3$ and $\gamma = 2$ achieved the best performance with an F_1-score of 0.976. For the multiclass image classifier model, which categorizes images into one of six defect categories, the ResNet-50 model trained with focal loss function using hyperparameter $\alpha = 1$ and $\gamma = 1$ achieved the best performance with an F_1-score of 0.976. In addition, we investigated both one-stage and two-stage object localization models to obtain more precise results, and our proposed methods based on the Faster R-CNN architecture with a ResNet-50 and VGG-19 feature extractor achieved an mAP of 0.87 and 0.83, respectively, on the testing set for ten defect categories.

In comparison, the detection system based on SSD architecture with a ResNet-50 and VGG-19 feature extractor achieved an mAP of 0.81 and 0.78, respectively. In general, the systems which use a ResNet feature extractor performed much better than those using a VGG feature extractor. We also investigated how the systems perform compared to a human examiner and found the Faster R-CNN system's performance to be highly comparable to manual inspection while taking ~35 minutes to assess a full cross-section image compared to ~1.5 hours for manual inspection (see Section 5.1.2). One major benefit of this data-driven approach is that it can be easily extended to more defect categories by simply including representative examples of the new defect categories in the training data and retraining the model. The results showed that transfer learning from a pre-trained model helps to speed up the training process and enables the model to generalize better even if the previous task is unrelated. This is especially important for defect detection tasks where a sufficiently large training dataset is often difficult and time-consuming to acquire. However, one limitation of this approach is that the model will likely need to be retrained for batteries with different chemistry and microstructures.

Besides various foreign inclusions and irregularities that the proposed model detected, one notable outcome of our experiments is that we discovered possible cross-contamination in the active material coating of the examined specimen of the Li-ion batteries, which would not have been possible otherwise. However, the evaluation of two cross-sections per battery cell is not sufficient for making a conclusive statement about the quality of the entire cell. Therefore, more cross-sections will need to be evaluated to make a statistically significant statement on the entire sample. Nevertheless, our proposed solution facilitates a new way of performing microscopy analysis for quality assessment. It enables automatic categorization, quantification, and visualization of regions of interest in a format (defect picture gallery) that allows the end-user to easily discover patterns from very large image data and draw insightful conclusions. For instance, it is not uncommon for materials scientists to look under the microscope to manually count the number of inclusions in a steel sample, which is extremely time-consuming for just one sample. With this approach, the purchasing or incoming goods department of an automotive manufacturer, for example, can compare the electrode quality of different production batches of a supplier or even compare the electrode quality of different manufacturers. This way, many more samples can be analysed in a few hours, and the detection system can be easily scaled by leveraging the vast amount of computing resources available in data centres. However, the complex electrode microstructure requires an elaborate preparation and image acquisition step to obtain a high-resolution image of the various components, which is a prerequisite for any quantitative or qualitative analysis. Nevertheless, the proposed solution can be deployed in addition to existing cell quality assessment methods.

Even with the successful results of the previously mentioned supervised learning approaches, the main drawback is the large number of labelled training samples required, which makes them impractical in certain situations. Improving the training data efficiencies and convergence capabilities of neural networks is an ongoing research area. However, unlike the models based on supervised learning, unsupervised models can discover previously unknown defect patterns in the data, making them particularly useful when properly designed. Although, anomaly detection in high-dimensional data remains a challenging and open research topic. Nevertheless, our proposed unsupervised method for detecting defects in sintered NdFeB magnets based on a variational autoencoder architecture achieved an F_1-score of 0.70 using a reconstruction threshold value of 0.69. However, due to the pixel-by-pixel loss, the output of the VAE model tends to be blurry and unspecific, resulting in a higher number of false positives depending on the selected threshold value. Subsequently, the system performance was improved by using a conditional generative adversarial network to reconstruct higher resolution output, which allows the decision threshold to be increased to 0.82. With this threshold value, we detected all the images with defects in the testing set without detecting any false positives. In addition, with the trained model, we could detect and compare the number of defects in a high quality magnet to those of a lower-grade magnet (see Section 4.4). This is of particular interest since all types of defects, especially soft magnetic phases, are nucleation points for domain reversal, resulting in lower coercivity and eventual demagnetisation. However, due to the various pre-processing and post-processing steps, the proposed unsupervised method is considerably slower (~2.5 hours per sample) than the supervised method. Furthermore, the unique contrast between the darker defects and the bright $Nd_2Fe_{14}B$ phase components in the microstructure of the magnet is particularly suited for the proposed unsupervised method. For example, with this approach, detecting defects such as a secondary phase and regions with large grains would be challenging since they have very similar appearances to the main hard magnetic phase $Nd_2Fe_{14}B$. In addition, such an approach would also not be successful in detecting defects in the microstructure of Li-ion ion batteries since the foreign inclusions and active material components often have very similar features. Lastly, a high-quality sample preparation procedure is of utmost importance for a reliable analysis using the proposed methods as it can often be difficult to differentiate real defects from preparation artefacts.

6.3 Impact of machine learning on microscopy

Advances in imaging technologies have enabled the acquisition of large volumes of microscopy images and made it possible to conduct large-scale, image-based experiments in various fields, from bioinformatics to material science. Computer vision and machine learning have a huge potential in automating the analysis and understanding of such large datasets. Compared to other domains involving images, the major advantage of using machine learning for microscopy image analysis is that microscopy provides a very controlled environment where performance can be showcased by using a carefully prepared training data set. As a result, the analysis can focus solely on the tasks the model is trained for without interference from unexpected external factors. Compared with traditional microscopy solutions, like deconvolution and convex optimisation, for example, once a machine learning model is trained, the inference is non-iterative and very fast to compute without the need for any parameter optimisations, even on modest computers and processors. Therefore, considering the current growth of tools employing machine learning in analytical microscopy, it is not difficult to predict that this trend will continue to expand further with machine learning used to tackle more ambitious tasks and the majority of manually programmed pipelines being gradually replaced with automated analytical solutions. For example, an automated process could convert either single micrographs or a series of micrographs into a microstructure representation that is easy to store, search and mine for feature similarity. Consequently, this will introduce new approaches that will increase processing output and accelerate new scientific discoveries. However, since most of the major advances in computer vision are happening in the field of computer science, it is often very difficult for experts in other fields to reproduce the same results. Therefore, computer scientists would need to work closely with domain experts such as material scientists or biologists since they have a better knowledge of which problems actually need solving. In addition, domain experts can provide a greater context to a problem that cannot necessarily be determined by simply accessing the data sets from a public repository without relevant background. Lastly, it is important to emphasize that the proposed systems are not a replacement for scientists but instead are meant to support scientists with various assistive functionalities in getting their work done.

6.4 Future work

As we become more cognizant of the limitations of our proposed methods, we hope that more work will be done to make them more robust. For example, performance evaluation on specimens from other research groups should be investigated to test the model robustness since it is common for deep learning models to rely on the specific imaging condition of a particular environment. In an ideal scenario, a dedicated generalisation set should be created in addition to a random test set when creating a new dataset. This generalization set will contain examples collected under various conditions from various research groups using different microscopes and settings. This will enable a thorough evaluation of how well our models generalize. Another important future research direction could be on how to improve the data efficiency of these models. Our proposed methods benefited considerably from transfer learning of pre-trained models. Therefore, we can imagine that specialized pre-training tasks that encode specific knowledge into the model will dramatically reduce the required amount of labelled training data and make developing better defect detection models more feasible. Furthermore, in the long run, other downstream models will benefit from a wide range of pre-trained representations either based on the cell chemistry or sample preparation method, making the approach more computationally efficient than starting from scratch. In addition, since acquiring representative training data is likely to remain a major bottleneck for developing accurate defect detection models, further development on unsupervised and self-supervised methods will be needed to deal with such low-resource scenarios. As we have shown, current unsupervised methods are still less performant than supervised approaches; thus, developing more robust unsupervised methods is an important research direction. For example, recent studies on texture and style transfer [161], [206] have shown that the hidden representations of a convolutional network can capture various spatial correlation properties in an image. Therefore, instead of direct pixel-wise subtraction of the original and reconstructed image, the subtraction can be carried out in the feature space of a pre-trained deep convolutional network.

Lastly, what is even more important in future experiments will be to evaluate the effects of the detected defects on the overall performance of the cells and magnets, which will help make more informed decisions. For instance, how many square millimetres of the cross-sections must be analysed in order to make conclusions about the total sample volume. However, such activity will require many specimens and cross-sections to be evaluated and quantified in a reasonable amount of time. Hence, an automated assessment of these samples using our proposed methods, for instance, is a prerequisite for such analyses.

Bibliography

[1] X. Xie, “A Review of Recent Advances in Surface Defect Detection using Texture analysis Techniques,” *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, vol. 7, no. 3, pp. 1–22, 2008.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[3] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, “Handwritten Digit Recognition with a Back-Propagation Network,” in *Advances in Neural Information Processing Systems 2*, 1990, pp. 396–404.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[5] V. Badrinarayanan, Kendall Alex, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[6] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[7] F. Liu, C. Shen, G. Lin, and I. Reid, “Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2024–2039, 2016.

[8] M. Kim, M. Lee, M. An, and H. Lee, “Effective automatic defect classification process based on CNN with stacking ensemble model for TFT-LCD panel,” *Journal of Intelligent Manufacturing*, vol. 31, no. 5, pp. 1165–1174, 2020.

[9] M. Ferguson, R. Ak, Y.-T. T. Lee, and K. H. Law, “Automatic localization of casting defects with convolutional neural networks,” in *2017 IEEE International Conference on Big Data*, 2017, pp. 1726–1735.

[10] B. Staar, M. Lütjen, and M. Freitag, “Anomaly detection with convolutional neural networks for industrial surface inspection,” *Procedia CIRP*, vol. 79, pp. 484–489, 2019.

[11] N. Nitta, F. Wu, J. T. Lee, and G. Yushin, “Li-ion battery materials: present and future,” *Materials Today*, vol. 18, no. 5, pp. 252–264, 2015.

[12] M. Armand and J.-M. Tarascon, “Building better batteries,” *Nature*, vol. 451, no. 7179, pp. 652–657, 2008.

[13] I. A. Jimenez Gordon, S. Grugeon, H. Takenouti, B. Tribollet, M. Armand, C. Davoisne, A. Debart, and S. Laruelle, “Electrochemical Impedance Spectroscopy response study of a commercial graphite-based negative electrode for Li-ion batteries as function of the

cell state of charge and ageing," *Electrochimica Acta*, vol. 223, pp. 63–73, 2017.

[14] M. Berecibar, I. Gandiaga, I. Villarreal, N. Omar, J. Van Mierlo, and P. den Bossche, "Critical review of state of health estimation methods of Li-ion batteries for real applications," *Renewable and Sustainable Energy Reviews*, vol. 56. pp. 572–587, 2016.

[15] Y. Li, M. Abdel Monem, R. Gopalakrishnan, M. Berecibar, E. Nanini-Maury, N. Omar, P. den Bossche, and J. Van Mierlo, "A quick on-line state of health estimation method for Li-ion battery with incremental capacity curves processed by Gaussian filter," *Journal of Power Sources*, vol. 373, pp. 40–53, 2018.

[16] Y. Wu, S. Saxena, Y. Xing, Y. Wang, C. Li, W. K. C. Yung, and M. Pecht, "Analysis of Manufacturing-Induced Defects and Structural Deformations in Lithium-Ion Batteries Using Computed Tomography," *Energies*, vol. 11, no. 4, pp. 1–22, 2018.

[17] T. Waldmann, A. Iturrondobeitia, M. Kasper, N. Ghanbari, F. Aguesse, E. Bekaert, L. Daniel, S. Geniès, I. J. Gordon, M. Löble, E. D. Vito, and M. Wohlfahrt-Mehrens, "Review—Post-Mortem Analysis of Aged Lithium-Ion Batteries: Disassembly Methodology and Physico-Chemical Analysis Techniques," *Journal of The Electrochemical Society*, vol. 163, no. 10, pp. A2149–A2164, 2016.

[18] L. Liang, T. Ma, P. Zhang, J. Jin, and M. Yan, "Coercivity enhancement of NdFeB sintered magnets by low melting point Dy32.5Fe62Cu5.5 alloy modification," *Journal of Magnetism and Magnetic Materials*, vol. 355, pp. 131–135, 2014.

[19] T. G. Woodcock, Y. Zhang, G. Hrkac, G. Ciuta, N. M. Dempsey, T. Schrefl, O. Gutfleisch, and D. Givord, "Understanding the microstructure and coercivity of high performance NdFeB-based magnets," *Scripta Materialia*, vol. 67, no. 6, pp. 536–541, 2012.

[20] K. Hono and H. Sepehri-Amin, "Strategy for high-coercivity Nd–Fe–B magnets," *Scripta Materialia*, vol. 67, no. 6, pp. 530–535, 2012.

[21] G. Sarriegui, J. M. Martín, M. Ipatov, A. P. Zhukov, and J. Gonzalez, "Magnetic Properties of NdFeB Alloys Obtained by Gas Atomization Technique," *IEEE Transactions on Magnetics*, vol. 54, no. 11, pp. 1–5, 2018.

[22] S. Gorse, B. Kugler, T. Samtleben, T. Waldmann, M. Wohlfahrt-Mehrens, G. Schneider, and V. Knoblauch, "An Explanation of the Ageing Mechanism of Li-Ion Batteries by Metallographic and Material Analysis," *Practical Metallography*, vol. 51, no. 12, pp. 829–848, 2014.

[23] B. Scrosati and J. Garche, "Lithium batteries: Status, prospects and future," *Journal of Power Sources*, vol. 195, no. 9, pp. 2419–2430, 2010.

[24] M. Perrin, Y. M. Saint-Drenan, F. Mattera, and P. Malbranche, "Lead–acid batteries in stationary applications: competitors and new markets for large penetration of renewable energies," *Journal of Power Sources*, vol. 144, no. 2, pp. 402–410, 2005.

[25] J. M. Tarascon and M. Armand, "Issues and challenges facing rechargeable lithium batteries," *Nature*, vol. 414, no. 6861, pp. 359–367, 2001.

[26] T. Gao and W. Lu, "Machine learning toward advanced energy storage devices and

systems," *iScience*, vol. 24, no. 1, pp. 1–33, 2021.

[27] C. Weisenberger, G. Guth, T. Bernthaler, and V. Knoblauch, "New Quality Evaluation Approaches for Lithium Ion Batteries Using the Interference Layer Metallography in Combination with Quantitative Structural Analysis," *Practical Metallography*, vol. 51, no. 1, pp. 5–31, 2014.

[28] M. Broussely, S. Herreyre, P. Biensan, P. Kasztejna, K. Nechev, and R. J. Staniewicz, "Aging mechanism in Li ion cells and calendar life predictions," *Journal of Power Sources*, vol. 97–98, pp. 13–21, 2001.

[29] J. Vetter, P. Novák, M. R. Wagner, C. Veit, K.-C. Möller, J. O. Besenhard, M. Winter, M. Wohlfahrt-Mehrens, C. Vogler, and A. Hammouche, "Ageing mechanisms in lithium-ion batteries," *Journal of Power Sources*, vol. 147, no. 1, pp. 269–281, 2005.

[30] J. B. Robinson, E. Engebretsen, D. P. Finegan, J. Darr, G. Hinds, P. R. Shearing, and D. J. L. Brett, "Detection of Internal Defects in Lithium-Ion Batteries Using Lock-in Thermography," *ECS Electrochemistry Letters*, vol. 4, no. 9, pp. A106–A109, 2015.

[31] J. Li, E. Murphy, J. Winnick, and P. A. Kohl, "Studies on the cycle life of commercial lithium ion batteries during rapid charge–discharge cycling," *Journal of Power Sources*, vol. 102, no. 1, pp. 294–301, 2001.

[32] M. Ecker, N. Nieto, S. Käbitz, J. Schmalstieg, H. Blanke, A. Warnecke, and D. U. Sauer, "Calendar and cycle life study of Li(NiMnCo)O2-based 18650 lithium-ion batteries," *Journal of Power Sources*, vol. 248, pp. 839–851, 2014.

[33] M. D. R. Kok, J. B. Robinson, J. S. Weaving, A. Jnawali, M. Pham, F. Iacoviello, D. J. L. Brett, and P. R. Shearing, "Virtual unrolling of spirally-wound lithium-ion cells for correlative degradation studies and predictive fault detection," *Sustainable Energy and Fuels*, vol. 3, no. 11, pp. 2972–2976, 2019.

[34] D. Mohanty, E. Hockaday, J. Li, D. K. Hensley, C. Daniel, and D. L. Wood, "Effect of electrode manufacturing defects on electrochemical performance of lithium-ion batteries: Cognizance of the battery failure sources," *Journal of Power Sources*, vol. 312, pp. 70–79, 2016.

[35] L. Yang, H. Sen Chen, W. L. Song, and D. Fang, "Effect of Defects on Diffusion Behaviors of Lithium-Ion Battery Electrodes: In Situ Optical Observation and Simulation," *ACS Applied Materials and Interfaces*, vol. 10, no. 50, pp. 43623–43630, 2018.

[36] J. B. Robinson, M. Maier, G. Alster, T. Compton, D. J. L. Brett, and P. R. Shearing, "Spatially resolved ultrasound diagnostics of Li-ion battery electrodes," *Physical Chemistry Chemical Physics*, vol. 21, pp. 6354–6361, 2019.

[37] N. Watrin, B. Blunier, and A. Miraoui, "Review of adaptive systems for lithium batteries State-of-Charge and State-of-Health estimation," in *2012 IEEE Transportation Electrification Conference and Expo (ITEC)*, 2012, pp. 1–6.

[38] D. P. Finegan, E. Darcy, M. Keyser, B. Tjaden, T. M. M. Heenan, R. Jervis, J. J. Bailey, N. T. Vo, O. V Magdysyuk, M. Drakopoulos, M. Di Michiel, A. Rack, G. Hinds, D. J. L. Brett, and P. R. Shearing, "Identifying the Cause of Rupture of Li-Ion Batteries during

Thermal Runaway,” *Advanced Science*, vol. 5, no. 1, pp. 1–13, 2018.

[39] D. P. Finegan, M. Scheel, J. B. Robinson, B. Tjaden, I. Hunt, T. J. Mason, J. Millichamp, M. Di Michiel, G. J. Offer, G. Hinds, D. J. L. Brett, and P. R. Shearing, “In-operando high-speed tomography of lithium-ion batteries during thermal runaway,” *Nature Communications*, vol. 6, no. 1, pp. 1–10, 2015.

[40] T. B. Reddy, *Linden’s handbook of batteries*, 4th ed. New York: McGraw-Hill Education, 2011.

[41] D. Choi, W. Wang, and Z. Yang, “Material Challenges and Perspectives,” in *Lithium-Ion Batteries: Advanced Materials and Technologies*, X. Yuan, H. Liu, and J. Zhang, Eds. Boca Raton: CRC Press, 2012, pp. 1–50.

[42] M. Schmitt, M. Baunach, L. Wengeler, K. Peters, P. Junges, P. Scharfer, and W. Schabel, “Slot-die processing of lithium-ion battery electrodes—Coating window characterization,” *Chemical Engineering and Processing: Process Intensification,* vol. 68, pp. 32–37, 2013.

[43] G. Qian, F. Monaco, D. Meng, S.-J. Lee, G. Zan, J. Li, D. Karpov, S. Gul, D. Vine, B. Stripe, J. Zhang, J.-S. Lee, Z.-F. Ma, W. Yun, P. Pianetta, X. Yu, L. Li, P. Cloeten, and Y. Liu, “The role of structural defects in commercial lithium-ion batteries,” *Cell Reports Physical Science*, vol. 2, no. 9, pp. 1–11, 2021.

[44] R. R. Richardson, C. R. Birkl, M. A. Osborne, and D. A. Howey, “Gaussian Process Regression for In Situ Capacity Estimation of Lithium-Ion Batteries,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 127–138, 2019.

[45] G. O. Sahinoglu, M. Pajovic, Z. Sahinoglu, Y. Wang, P. V Orlik, and T. Wada, “Battery State-of-Charge Estimation Based on Regular/Recurrent Gaussian Process Regression,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4311–4321, 2018.

[46] Y. Li, C. Zou, M. Berecibar, E. Nanini-Maury, J. C.-W. Chan, P. van den Bossche, J. Van Mierlo, and N. Omar, “Random forest regression for online capacity estimation of lithium-ion batteries,” *Applied Energy*, vol. 232, pp. 197–210, 2018.

[47] E. Chemali, P. J. Kollmeyer, M. Preindl, and A. Emadi, “State-of-charge estimation of Li-ion batteries using deep neural networks: A machine learning approach,” *Journal of Power Sources*, vol. 400, pp. 242–255, 2018.

[48] C. Liu, J. Tan, H. Shi, and X. Wang, “Lithium-Ion Cell Screening With Convolutional Neural Networks Based on Two-Step Time-Series Clustering and Hybrid Resampling for Imbalanced Data,” *IEEE Access*, vol. 6, pp. 59001–59014, 2018.

[49] K. A. Severson, P. M. Attia, N. Jin, N. Perkins, B. Jiang, Z. Yang, M. H. Chen, M. Aykol, P. K. Herring, D. Fraggedakis, M. Z. Bazant, S. J. Harris, W. C. Chueh, and R. D. Braatz, “Data-driven prediction of battery cycle life before capacity degradation,” *Nature Energy*, vol. 4, pp. 383–391, 2019.

[50] Y. Zhou, M. Huang, and M. Pecht, “Remaining useful life estimation of lithium-ion cells based on k-nearest neighbor regression with differential evolution optimization,” *Journal of Cleaner Production*, vol. 249, pp. 1–12, 2020.

[51] Y. Zhou, M. Huang, Y. Chen, and Y. Tao, “A novel health indicator for on-line lithium-ion batteries remaining useful life prediction,” *Journal of Power Sources*, vol. 321, pp. 1–10, 2016.

[52] J. P. Ortiz, J. D. Valladolid, C. L. Garcia, G. Novillo, and F. Berrezueta, “Analysis of Machine Learning Techniques for the Intelligent Diagnosis of Ni-MH Battery Cells,” in *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, 2018, pp. 1–6.

[53] S. N. Haider, Q. Zhao, and X. Li, “Data driven battery anomaly detection based on shape based clustering for the data centers class,” *Journal of Energy Storage*, vol. 29, pp. 1–10, 2020.

[54] X. Li, T. Zhang, and Y. Liu, “Detection of Voltage Anomalies in Spacecraft Storage Batteries Based on a Deep Belief Network,” *Sensors*, vol. 19, no. 21, pp. 1–21, 2019.

[55] L. Yao, Y. Xiao, X. Gong, J. Hou, and X. Chen, “A novel intelligent method for fault diagnosis of electric vehicle battery system based on wavelet neural network,” *Journal of Power Sources*, vol. 453, pp. 1–12, 2020.

[56] Y. Wang, Y. Chen, X. Liao, and L. Dong, “Lithium-ion Battery Face Imaging with Contactless Walabot and Machine Learning,” in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2019, pp. 1067–1072.

[57] O. Badmos, A. Kopp, T. Bernthaler, and G. Schneider, “Image-based defect detection in lithium-ion battery electrode using convolutional neural networks,” *Journal of Intelligent Manufacturing*, vol. 31, no. 4, pp. 885–897, 2020.

[58] L. Ma, W. Xie, and Y. Zhang, “Blister Defect Detection Based on Convolutional Neural Network for Polymer Lithium-Ion Battery,” *Applied Sciences*, vol. 9, no. 6, pp. 1–15, 2019.

[59] X. Tang, K. Yao, B. Liu, W. Hu, and F. Gao, “Long-Term Battery Voltage, Power, and Surface Temperature Prediction Using a Model-Based Extreme Learning Machine,” *Energies*, vol. 11, no. 1, pp. 1–16, 2018.

[60] Z. Zheng, B. Chen, Y. Xu, N. Fritz, Y. Gurumukhi, J. Cook, M. N. Ates, N. Miljkovic, P. V Braun, and P. Wang, “A Gaussian Process-Based Crack Pattern Modeling Approach for Battery Anode Materials Design,” *Journal of Electrochemical Energy Conversion and Storage*, vol. 18, no. 1, pp. 1–9, 2020.

[61] Y. Takagishi, T. Yamanaka, and T. Yamaue, “Machine Learning Approaches for Designing Mesoscale Structure of Li-Ion Battery Electrodes,” *Batteries*, vol. 5, no. 3, pp. 1–14, 2019.

[62] B. Wu, S. Han, K. G. Shin, and W. Lu, “Application of artificial neural networks in design of lithium-ion batteries,” *Journal of Power Sources*, vol. 395, pp. 128–136, 2018.

[63] T. Gao and W. Lu, “Physical Model and Machine Learning Enabled Electrolyte Channel Design for Fast Charging,” *Journal of The Electrochemical Society*, vol. 167, no. 11, pp. 1–21, 2020.

[64] R. R. Richardson, M. A. Osborne, and D. A. Howey, “Gaussian process regression for

forecasting battery state of health," *Journal of Power Sources*, vol. 357, pp. 209–219, 2017.

[65] M. Sagawa, S. Fujimura, N. Togawa, H. Yamamoto, and Y. Matsuura, "New material for permanent magnets on a base of Nd and Fe," *Journal of Applied Physics*, vol. 55, no. 6, pp. 2083–2087, 1984.

[66] D. J. Branagan, "A metallurgical approach toward alloying in rare earth permanent magnet systems," doctoral dissertation, Ames Laboratory, Ames, IA, USA, 1995.

[67] N. T. Oster, "Generation and characterization of anisotropic microstructures in rare earth-iron-boron alloys," doctoral dissertation, Department of Materials Science and Engineering, Iowa State University, Ames, IA, USA, 2012.

[68] J. F. Gieras, R.-J. Wang, and M. J. Kamper, *Axial flux permanent magnet brushless machines*, 2nd ed. Dordrecht: Springer, 2008.

[69] J. F. Gieras, *Permanent Magnet Motor Technology: Design and Applications*, 2nd ed. Boca Raton: CRC Press, 2002.

[70] S. Kruse, K. Raulf, A. Trentmann, T. Pretz, and B. Friedrich, "Processing of Grinding Slurries Arising from NdFeB Magnet Production," *Chemie Ingenieur Technik*, vol. 87, no. 11, pp. 1589–1598, 2015.

[71] J. Lucas, P. Lucas, T. Le Mercier, A. Rollat, and W. G. Davenport, *Rare earths: science, technology, production and use*. Amsterdam: Elsevier, 2014.

[72] N. Krishnamurthy and C. K. Gupta, *Extractive metallurgy of rare earths*, 2nd ed. Boca Raton: CRC Press, 2015.

[73] H. Kronmuller and K. D. Durst, "Magnetic Hardening Mechanism in Re-Fe-B Permanent Magnets," in *Concerted European Action on Magnets (CEAM)*, I. V. Mitchell, J. M. D. Coey, D. Givord, I. R. Harris, and R. Hanitsch, Eds. Dordrecht: Springer, 1989, pp. 392–404.

[74] T.-W. Chang, K.-W. Liao, C.-C. Lin, M.-C. Tsai, and C.-W. Cheng, "Predicting magnetic characteristics of additive manufactured soft magnetic composites by machine learning," *The International Journal of Advanced Manufacturing Technology*, vol. 114, no. 9, pp. 3177–3184, 2021.

[75] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[76] R. Li, Y. Liu, S.-L. Zuo, T.-Y. Zhao, F.-X. Hu, J.-R. Sun, and B.-G. Shen, "Composition design for (PrNd-La-Ce)2Fe14B melt-spun magnets by machine learning technique," *Chinese Physics B*, vol. 27, no. 4, pp. 1–5, 2018.

[77] L. Exl, J. Fischbacher, A. Kovacs, H. Özelt, M. Gusenbauer, K. Yokota, T. Shoji, G. Hrkac, and T. Schrefl, "Magnetic microstructure machine learning analysis," *Journal of Physics: Materials*, vol. 2, no. 1, pp. 1–19, 2018.

[78] X. Tang, J. Li, A. K. Srinithi, H. Sepehri-Amin, T. Ohkubo, and K. Hono, "Role of V

on the coercivity of SmFe12-based melt-spun ribbons revealed by machine learning and microstructure characterizations," *Scripta Materialia*, vol. 200, pp. 1–5, 2021.

[79] H.-K. Park, J.-H. Lee, J. Lee, and S.-K. Kim, "Optimizing machine learning models for granular NdFeB magnets by very fast simulated annealing," *Scientific Reports*, vol. 11, no. 1, pp. 1–11, 2021.

[80] M. Gusenbauer, H. Oezelt, J. Fischbacher, A. Kovacs, P. Zhao, T. G. Woodcock, and T. Schrefl, "Extracting local nucleation fields in permanent magnets using machine learning," *npj Computational Materials*, vol. 6, no. 1, pp. 1–10, 2020.

[81] A. K. Choudhary, "Machine learning for microstructures classification in functional materials," Master's thesis, Department of Materials Science, Aalen University, Aalen, Germany, 2018.

[82] T. R. Reed and J. M. H. Dubuf, "A Review of Recent Texture Segmentation and Feature Extraction Techniques," *CVGIP: Image Understanding*, vol. 57, no. 3, pp. 359–372, 1993.

[83] H. Y. T. Ngan, G. K. H. Pang, and N. H. C. Yung, "Automated fabric defect detection-A review," *Image and Vision Computing*, vol. 29, no. 7, pp. 442–458, 2011.

[84] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[85] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992, pp. 144–152.

[86] N. Cristianini and E. Ricci, "Support Vector Machines," in *Encyclopedia of Algorithms*, Boston, MA: Springer, 2008, pp. 928–932.

[87] C. Mandriota, M. Nitti, N. Ancona, E. Stella, and A. Distante, "Filter-based feature selection for rail defect detection," *Machine Vision and Applications*, vol. 15, no. 4, pp. 179–185, 2004.

[88] V. Murino, M. Bicego, and I. A. Rossi, "Statistical classification of raw textile defects," in *Proceedings of the 17th International Conference on Pattern Recognition ICPR*, 2004, vol. 4, pp. 311–314.

[89] M. P. Samy, S. Foong, G. S. Soh, and K. S. Yeo, "Automatic optical laser-based defect detection and classification in brick masonry walls," in *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 3521–3524.

[90] Z. Xue, Y. Shang, and A. Feng, "Semi-supervised outlier detection based on fuzzy rough C-means clustering," *Mathematics and Computers in Simulation*, vol. 80, no. 9, pp. 1911–1921, 2010.

[91] F. Ramírez and H. Allende, "Detection of flaws in aluminium castings: A comparative study between generative and discriminant approaches," *Insight - Non-Destructive Testing and Condition Monitoring*, vol. 55, pp. 366–371, 2013.

[92] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion

detection: A review," *Computers & Security*, vol. 30, no. 6, pp. 353–375, 2011.

[93] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.

[94] C. Brekke and A. H. S. Solberg, "Oil spill detection by satellite remote sensing," *Remote Sensing of Environment*, vol. 95, no. 1, pp. 1–13, 2005.

[95] F. Ramírez, H. Allende-Cid, A. Veloz, and H. Allende, "Neuro-fuzzy-based Arrhythmia Classification Using Heart Rate Variability Features," in *2010 XXIX International Conference of the Chilean Computer Science Society*, 2010, pp. 205–211.

[96] D. M. J. Tax, "One-class classification: Concept learning in the absence of counter-examples," Ph.D. dissertation, Technische Universiteit Delft, Delft, 2001.

[97] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[98] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A Novel Anomaly Detection Scheme Based on Principal Component Classifier," in *3rd IEEE International Conference on Data Mining*, 2003, pp. 353–365.

[99] H. Liu, Z. Ma, S. Zhang, and X. Wu, "Penalized partial least square discriminant analysis with ℓ1-norm for multi-label data," *Pattern Recognition*, vol. 48, no. 5, pp. 1724–1733, 2015.

[100] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. van den Hengel, "Memorizing Normality to Detect Anomaly: Memory-Augmented Deep Autoencoder for Unsupervised Anomaly Detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1705–1714.

[101] V. Nair and G. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair," in *Proceedings of ICML*, 2010, vol. 27, pp. 807–814.

[102] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[103] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[104] L. Smith, "Cyclical learning rates for training neural networks," in *IEEE Winter Conference on Applications of Computer Vision, WACV*, 2017, pp. 464–472.

[105] L. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Proceedings SPIE, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019, vol. 11006, pp. 369–386.

[106] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o(1/k^2)," *Doklady AN USSR*, vol. 269, no. 3, pp. 543–547, 1983.

[107] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-print arXiv:1412.6980*, 2014.

[108] C. Zhang, K. C. Tan, H. Li, and G. S. Hong, "A Cost-Sensitive Deep Belief Network for Imbalanced Classification," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 30, no. 1, pp. 109–122, 2019.

[109] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence),* 2008, pp. 1322–1328.

[110] P. Lim, C. K. Goh, and K. C. Tan, "Evolutionary Cluster-Based Synthetic Oversampling Ensemble (ECO-Ensemble) for Imbalance Learning," *IEEE Transactions on Cybernetics,* vol. 47, no. 9, pp. 2850–2861, 2017.

[111] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[112] S. García and F. Herrera, "Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy," *Evolutionary Computation*, vol. 17, no. 3, pp. 275–306, 2009.

[113] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[114] C. L. Castro and A. P. Braga, "Novel Cost-Sensitive Approach to Improve the Multilayer Perceptron Performance on Imbalanced Data," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 24, no. 6, pp. 888–899, 2013.

[115] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.

[116] S. Datta and S. Das, "Near-Bayesian Support Vector Machines for imbalanced data classification with equal or unequal misclassification costs," *Neural Networks,* vol. 70, pp. 39–52, 2015.

[117] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, "Training deep neural networks on imbalanced data sets," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 4368–4374.

[118] S. Yue and T. Wang, "Imbalanced Malware Images Classification: a CNN based Approach," *arXiv e-print arXiv:1708.08042*, 2017.

[119] B. K. Baloch, S. Kumar, S. Haresh, A. Rehman, and T. Syed, "Focused Anchors Loss: cost-sensitive learning of discriminative features for imbalanced classification," in *Proceedings of The Eleventh Asian Conference on Machine Learning,* 2019, vol. 101, pp. 822–835.

[120] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 42, no. 2, pp. 318–327, 2020.

[121] Y. Ma, J. Sun, Q. Zhou, K. Cheng, X. Chen, and Y. Zhao, “CHS-NET: A Cascaded Neural Network with Semi-Focal Loss for Mitosis Detection,” in *Proceedings of The 10th Asian Conference on Machine Learning*, 2018, vol. 95, pp. 161–175.

[122] K. Doi and A. Iwasaki, “The Effect of Focal Loss in Semantic Segmentation of High Resolution Aerial Image,” in *IEEE International Geoscience and Remote Sensing Symposium*, 2018, pp. 6919–6922.

[123] D. H. Hubel and T. N. Wiesel, “Receptive fields of single neurones in the cat’s striate cortex,” *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[124] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[125] K. Fukushima, “Neocognitron: A hierarchical neural network capable of visual pattern recognition,” *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.

[126] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[127] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” *Computer Vision – ECCV 2014. Lecture Notes in Computer Science*, vol. 8689, pp. 818–833, 2014.

[128] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[129] W. Shi, J. Caballero, L. Theis, F. Huszar, A. Aitken, C. Ledig, and Z. Wang, “Is the deconvolution layer the same as a convolutional layer?,” *arxiv e-print arXiv:1609.07009*, 2016.

[130] D. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” *arXiv e-print arXiv:1312.6114*, 2013.

[131] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic Backpropagation and Approximate Inference in Deep Generative Models,” in *Proceedings of the 31st International Conference on Machine Learning*, 2014, vol. 32, no. 2, pp. 1278–1286.

[132] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[133] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2014, pp. 2672–2680.

[134] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” *arXiv e-print arXiv:1511.06434*, 2015.

[135] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” *arXiv e-print arXiv:1411.1784*, 2014.

[136] J.-Y. Zhu, T. Park, P. Isola, and A. Efros, “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks,” in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251.

[137] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 2180–2188.

[138] Y. Yang, X. Ni, Y. Hao, C. Liu, W. Wang, Y. Liu, and H. Xie, “MF-GAN: Multi-conditional Fusion Generative Adversarial Network for Text-to-Image Synthesis,” in *MultiMedia Modeling. MMM 2022. Lecture Notes in Computer Science*, vol. 13141, Cham: Springer, 2022, pp. 41–53.

[139] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, “GauGAN,” in *ACM SIGGRAPH 2019 Real-Time Live!*, 2019, pp. 1–1.

[140] H. Wang, W. Wu, Y. Su, Y. Duan, and P. Wang, “Image Super-Resolution using a Improved Generative Adversarial Network,” in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2019, pp. 312–315.

[141] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,” in *International Conference on Learning Representations (ICLR)*, 2014, pp. 1–16.

[142] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *arXiv e-print arXiv:1311.2524*, 2013.

[143] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, “Selective Search for Object Recognition,” *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[144] C. L. Zitnick and P. Dollár, “Edge Boxes: Locating Object Proposals from Edges,” in *Computer Vision - ECCV*, 2014, pp. 391–405.

[145] S. Shah, “Automatic object detection using objectness measure,” in *1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, 2013, pp. 1–6.

[146] J. Carreira and C. Sminchisescu, “CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1312–1328, 2012.

[147] I. Endres and D. Hoiem, “Category Independent Object Proposals,” in *Computer Vision - ECCV*, 2010, pp. 575–588.

[148] P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, “Multiscale Combinatorial Grouping,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 328–335.

[149] R. Girshick, “Fast R-CNN,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[150] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[151] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector,” in *Computer Vision - ECCV*, 2016, pp. 21–37.

[152] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv e-print arXiv:1409.1556*, 2014.

[153] J. Niedermeier, “Methoden zur Bewertung der Qualität von Lithium-Ionen-Batterie-Elektroden,” Master’s thesis, Department of Materials Science, Aalen University, Aalen, Germany, 2019.

[154] F. Trier, “Simple Big Image (SBI) version 2.0 [Computer software],” 2019. https://github.com/Materials-Research-Institute-Aalen/SimpleBigImage (accessed Aug. 23, 2022).

[155] J. Davis and M. Goadrich, “The Relationship between Precision-Recall and ROC Curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 233–240.

[156] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, vol. 9, pp. 249–256.

[157] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[158] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in *Computer Vision - ECCV*, 2014, pp. 740–755.

[159] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[160] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” in *Proceedings 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.

[161] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual Losses for Real-Time Style Transfer and Super-Resolution,” in *Computer Vision - ECCV*, 2016, pp. 694–711.

[162] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon, “Pixel-Level Domain Transfer,” in *Computer Vision - ECCV*, 2016, pp. 517–532.

[163] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context Encoders: Feature Learning by Inpainting,” in *IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR), 2016, pp. 2536–2544.

[164] X. Wang and A. Gupta, "Generative Image Modeling Using Style and Structure Adversarial Networks," in *Computer Vision - ECCV*, 2016, pp. 318–335.

[165] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Medical Image Computing and Computer-Assisted Intervention - MICCAI*, vol. 9351, pp. 234–241, 2015.

[166] M. Lin, Q. Chen, and S. Yan, "Network In Network," *arXiv e-print arXiv:1312.4400*, 2013.

[167] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2921–2929.

[168] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3296–3305.

[169] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper With Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[170] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[171] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, vol. 37, pp. 448–456.

[172] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, vol. 1524, G. Montavon, K. R. Müller, and G. B. Orr, Eds. Berlin: Springer, 1998, pp. 9–50.

[173] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv e-print arXiv:1312.6120*, 2013.

[174] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[175] K. He and J. Sun, "Convolutional Neural Networks at Constrained Time Cost," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5353–5360.

[176] N. D. Nguyen, T. Do, T. D. Ngo, and D. D. Le, "An Evaluation of Deep Learning Methods for Small Object Detection," *Journal of Electrical and Computer Engineering*, vol. 2020, pp. 1–18, 2020.

[177] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, “Object Detection With Deep Learning: A Review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[178] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, “SOD-MTGAN: Small Object Detection via Multi-Task Generative Adversarial Network,” in *Computer Vision - ECCV*, 2018, pp. 210–226.

[179] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.

[180] J. Friedman, “Greedy Function Approximation: A Gradient Boosting Machine,” *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[181] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, “Improving Variational Inference with Inverse Autoregressive Flow,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 4743–4751.

[182] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, “Semi-Supervised Learning with Deep Generative Models,” *arXiv e-print arXiv:1406.5298*, 2014.

[183] M. Arjovsky and L. Bottou, “Towards Principled Methods for Training Generative Adversarial Networks,” *arXiv e-print arXiv:1701.04862*, 2017.

[184] A. Bhattad, J. Rock, and D. Forsyth, “Detecting anomalous faces with ‘no peeking’ autoencoders,” *arXiv e-print arXiv:1802.05798*, 2018.

[185] L. Beggel, M. Pfeiffer, and B. Bischl, “Robust Anomaly Detection in Images Using Adversarial Autoencoders,” in *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2019. Lecture Notes in Computer Science*, vol. 11906, U. Brefeld, E. Fromont, A. Hotho, M. Maathuis, and C. Robardet, Eds. Cham: Springer, 2020, pp. 206–222.

[186] M. Mathieu, C. Couprie, and Y. LeCun, “Deep multi-scale video prediction beyond mean square error,” *arXiv e-print arXiv:1511.05440*, 2015.

[187] D. M. J. Tax and R. P. W. Duin, “Support Vector Data Description,” *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[188] D. Wang, D. S. Yeung, and E. C. C. Tsang, “Structured One-Class Classification,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 6, pp. 1283–1295, 2006.

[189] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2001.

[190] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[191] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying Density-Based Local Outliers,” in *Proceedings of the 2000 ACM SIGMOD International Conference*

on Management of Data, 2000, pp. 93–104.

[192] S. Abe, *Support vector machines for pattern classification*. London: Springer, 2005.

[193] L. Auria and R. A. Moro, "Support Vector Machines (SVM) as a Technique for Solvency Analysis," *SSRN Electronic Journal*, vol. 811, pp. 1–16, 2008.

[194] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep One-Class Classification," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, vol. 80, pp. 4393–4402.

[195] F. J. Huang and Y. LeCun, "Large-scale Learning with SVM and Convolutional for Generic Object Categorization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, vol. 1, pp. 284–291.

[196] D. Liu, H. Qian, G. Dai, and Z. Zhang, "An iterative SVM approach to feature selection and classification in high-dimensional datasets," *Pattern Recognition*, vol. 46, no. 9, pp. 2531–2537, 2013.

[197] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, pp. 217–288, 2011.

[198] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, no. 4, pp. 411–430, 2000.

[199] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[200] Y. Liu, S. Zhou, and Q. Chen, "Discriminative deep belief networks for visual data classification," *Pattern Recognition*, vol. 44, no. 10, pp. 2287–2296, 2011.

[201] L. J. Cao, K. S. Chua, W. K. Chong, H. P. Lee, and Q. M. Gu, "A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine," *Neurocomputing*, vol. 55, no. 1, pp. 321–336, 2003.

[202] S. Jahanbin, A. C. Bovik, E. Pérez, and D. Nair, "Automatic inspection of textured surfaces by support vector machines," in *Optical Inspection and Metrology for Non-Optics Industries*, 2009, vol. 7432, pp. 97–107.

[203] J. Andrews, E. Morton, and L. Griffin, "Detecting anomalous data using auto-encoders," *International Journal of Machine Learning and Computing*, vol. 6, no. 1, pp. 21–26, 2016.

[204] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[205] T. Ergen, A. H. Mirza, and S. S. Kozat, "Unsupervised and Semi-supervised Anomaly Detection with LSTM Neural Networks," *arXiv e-print arXiv:1710.09207*, 2017.

[206] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2414–2423.

List of publications

Peer-reviewed journal publications:

O. Badmos, A. Kopp, T. Bernthaler, and G. Schneider, “Image-based defect detection in lithium-ion battery electrode using convolutional neural networks,” J. Intell. Manuf., vol. 31, no. 4, pp. 885–897, 2020.

Conference contributions:

Application of deep learning in materials microscopy to evaluate Lithium-Ion batteries. Deutsche Gesellschaft für angewandte Optik, DGaO-Proceedings, July 2018.

Deep learning methods in microscopy for evaluating the quality of lithium-ion batteries. Materials Science and Engineering Congress, Darmstadt, Germany, September 2018.

Quality evaluation tool for automated defect detection in li-ion battery electrode using deep learning algorithms. International Battery Production Conference 2018, Braunschweig, Germany, November 2018.

Application of deep convolutional neural networks (DCNN) in materials microscopy for the automated detection of defects. Werkstoffwoche 2019, Dresden, Germany, September 2019.