Masaryk University
Faculty of Informatics

Defect detection in electron-optical aperture using neural networks

Master's Thesis

Bc. Štěpán Beneš

Brno, Fall 2020

Masaryk University
Faculty of Informatics

Defect detection in electron-optical aperture using neural networks

Master's Thesis

Bc. Štěpán Beneš

Brno, Fall 2020

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Bc. Štěpán Beneš

Advisor: doc. RNDr. Pavel Matula, Ph.D.

Acknowledgements

I'd like to hereby extend my gratitude to my supervisor doc. RNDr. Pavel Matula, Ph.D. for his valuable insights and guidance, shown faith, and perhaps most importantly his indefatigable optimism. I'd also like to thank Ing. Vojtěch Filip for clarification of all things related to scanning electron microscopes, and his efforts in providing additional data and detailed answers to my inquiries regarding defective apertures. Additionally, I thank TESCAN for an interesting topic to work on, and of course the received grants.

Secondly, I thank my family and close friends for their support during this strenuous year, as well as my dear Veronika "Tut" H. for proofreading services and morale-boosting.

I'd also like to mention Manchester United FC, for providing me with emotional roller-coaster during my studies, and for sometimes actually winning a game.

Finally, I'd like to thank myself, as it is quite certain that I could not have done this without me.

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

Abstract

The goal of this thesis was to explore a variety of neural network-based anomaly detection techniques and gauge their potential to be used as automatic defect detectors in a real-world manufacturing process. To this end a small data set consisting of images of predominantly non-defective electron-optical apertures was provided by TESCAN.

The thesis provides a thorough overview of neural network architectures applicable for the task of anomaly detection on image data. Six different approaches were chosen to be tested on the provided data set. Out of them, four were shown to be capable of reaching perfect anomaly detection abilities in some capacity.

The results imply that employing neural networks as automated defect detectors is indeed possible in this real-world scenario.

Keywords

Anomaly detection, Manufacturing defect detection, Machine learning, Neural networks, Computer vision, Image analysis, Automation

Contents

Introduction **1**

1 Anomaly detection **5**
1.1 Challenges 6
1.2 Nature of input data 7
1.3 Types of anomalies 9
1.4 Detection outputs 10
1.5 Surveys of anomaly detection methods 11

2 Deep learning methods for anomaly detection in image data **13**
2.1 Reconstruction-based methods 15
2.1.1 Autoencoders 15
2.1.2 Denoising autoencoders 17
2.1.3 Convolutional autoencoders 18
2.1.4 Recent applications 19
2.2 Distance metric-based methods 19
2.2.1 Siamese networks 20
2.2.2 Triplet networks 22
2.2.3 Recent applications 24
2.3 Hybrid models 25
2.3.1 One-class SVM 26
2.3.2 Recent applications 29
2.4 Generative model-using methods 29
2.4.1 Variational autoencoders 31
2.4.2 Generative adversarial networks 34
2.4.3 Wasserstein GAN 36
2.4.4 anoGAN 38
2.4.5 biGAN 40
2.4.6 Recent applications 41
2.5 One-class neural networks 41
2.5.1 Deep SVDD 42
2.5.2 Recent applications 44

3 Data set **45**
3.1 Types of defects 46
3.2 Challenges 52

4 Experiments **55**
4.1 Autoencoder approach 57
4.1.1 Pure AE 57
4.1.2 Ensemble with granulometry 60
4.2 Siamese network approach 63
4.3 Triplet network approach 66
4.4 Hybrid oc-SVM approach 67
4.5 Variational autoencoder approach 68
4.6 BiGAN approach 71

5 Results **77**
5.1 Autoencoder approach 77
5.1.1 Pure AE 77
5.1.2 Ensemble with granulometry 81
5.2 Siamese network approach 83
5.3 Triplet network approach 88
5.4 Hybrid oc-SVM approach 91
5.5 Variational autoencoder approach 94
5.6 BiGAN approach 99

6 Conclusion **103**

Bibliography **107**

A Electronic appendix overview **125**

B High-resolution images of apertures **127**

C Prototype apertures **147**

D Neural networks used in the hybrid approach **157**

Introduction

A common task when analysing real-world data sets is determining which data instances stand out as dissimilar from the rest. Such instances are known as *anomalies* (or *outliers*) and the goal of anomaly detection is to identify these instances in a data-driven fashion [1].

Anomaly detection has garnered much attention in diverse research and application domains, especially in areas where large data sets obtained from critical systems are concerned [2]. These domains include for example fraud detection [3, 4, 5], video surveillance [6, 7, 8], sensor networks [9, 10], cyber-intrusion detection [11, 12, 13], medical domain [14, 15, 16, 17, 18, 19] and industrial monitoring [20, 21, 22].

A sub-domain of aforementioned industrial monitoring, and one this thesis concerns itself with, is automated quality inspection in production processes. For industries, where safety is an utmost priority, e.g. medical or automobile, proper quality inspection is crucial. Manual inspection can be repetitive and prone to human error, but also infeasible in some settings due to production rates. Automatic inspection algorithms are still for a large part reliant on expert-crafted features [23, 24]. While these hand-crafted features may be powerful, their creation is laborious and might be necessary for each new product. A more general solution, that can automatically adapt to new problems, could therefore result in significant time and cost reduction [25]. Recent work of Pittino et al. [26] claims that despite the abundance of work on anomaly detection in an industrial setting, there is still a notable lack of application of anomaly detection methods on real world use-cases.

In the field of machine learning, recent years have seen the rise of deep learning across numerous domains. One such notable domain would be computer vision [27, 28, 29, 30]. Deep learning achieves its great performance and flexibility by learning to represent data as a hierarchy of concepts within layers of a neural network [31]. As such they are capable of extracting high-level features, and therefore no manual feature-crafting is needed.

Deep learning-based anomaly detection methods are becoming increasingly more popular and have been applied to a wide variety

of tasks[31]. Some studies even claim that deep learning-based methods have completely surpassed the traditional methods [32, 33, 31].

The goal of this thesis is to explore and compare modern neural networks-based anomaly detection methods on a provided data set, consisting of image data of electron-optical apertures, captured using scanning electron microscope.

The thesis is divided into six chapters, whose short **summary** follows:

- Chapter 1 serves as an introduction to the task of anomaly detection and its challenges.
- Chapter 2 is dedicated to a thorough overview of deep learning methods for anomaly detection on image data, consisting of providing an overall categorization of approaches, description of principles and neural network models used by each approach, and recent applications of each approach.
- Chapter 3 contains a detailed description of the provided data set.
- Chapter 4 describes the experimental settings for each tested approach and the overall performance evaluation criteria.
- Chapter 5 provides the evaluation of results, as well as insights into the tuning of the hyperparameters and choices of architectural features of each tested approach.
- Finally, Chapter 6 concludes the thesis with a comparison of the evaluated approaches.

Main contribution of this thesis lies in evaluating a significant number of modern deep learning-based anomaly detection methods on a real-world data set. Several of the implemented approaches have learned to distinguish defective (anomalous) apertures in a perfect manner, without any false negatives and false positives, and could therefore be considered as practically usable.

This thesis also tries to provide, as a byproduct, an up-to-date taxonomy of deep learning methods for anomaly detection on image data.

And while the task of creating a ready for use anomaly detector was not a goal of this thesis, the electronic appendix contains saved trained models and performance evaluating scripts, which could fairly easily be turned into usable detectors.

1 Anomaly detection

The task of anomaly detection refers to a problem of finding patterns in data that do not conform to expected behaviour. These non-conforming patterns are widely referred to as anomalies, outliers, aberrations or contaminants. Of these, anomalies and outliers are the terms most commonly used in the context of anomaly detection, oftentimes interchangeably [1]. Hawkins [34] provides another point of view when describing anomalies as an observation that deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism.

Another topic closely related to anomaly detection is novelty detection [35], which aims at detecting new, previously unseen patterns in the data. The main difference is that novelties are not considered as anomalous data points, and are usually incorporated into the normal model, once detected [31]. The techniques employed for novelty and anomaly detection are virtually the same however [36].

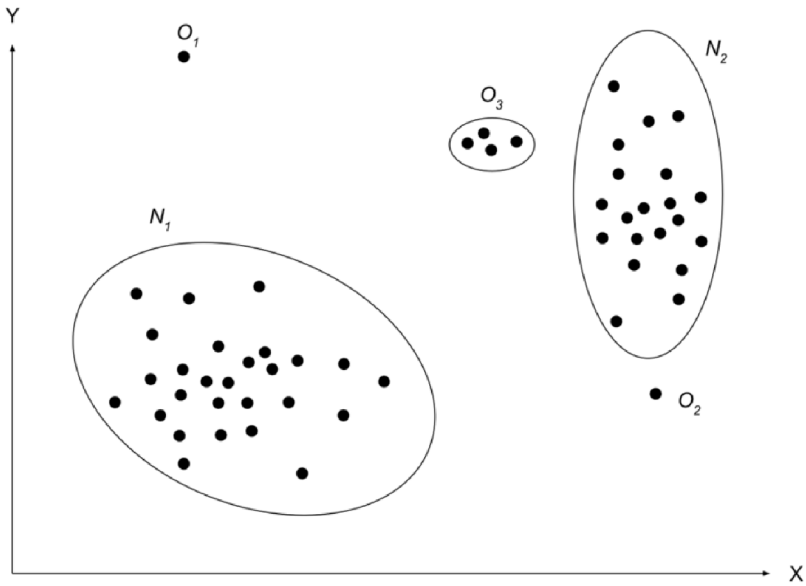


Figure 1.1: Illustration of anomalies in 2D data set, where N_1 and N_2 refer to normal data regions and O_1, O_2 and O_3 to outliers [1].

Anomalies might exist in the data for various reasons, such as malicious activity, cyber-intrusion or breakdown of a system. But no matter the cause, anomalies of all kinds share the characteristic of being interesting to the analyst. Indeed, the importance of anomaly detection lies in the fact that anomalies in data translate to significant, often even critical, actionable information in a wide spectrum of application domains [1].

1.1 Challenges

A straightforward anomaly detection approach would be to define a region representing normal behaviour and proclaim any observation in the data that would not belong to this region as an anomaly. Chandola et al. [1] describe several factors that make such approach problematic:

- Defining a normal region that would encompass every possible normal behaviour can be very difficult. In addition, the boundary between normal and anomalous is often not precise.
- In many domains, the notion of normal behaviour keeps evolving and might vastly differ in the future.
- Availability of labeled data for training or validation of models used by anomaly detection techniques is usually a major issue.
- Often the data contains noise, which tends to be similar to actual anomalies.

Due to the above-mentioned challenges, the anomaly detection problem in its most general form is not easy to solve. Most of the existing anomaly detection techniques solve a specific formulation of the problem, induced by various factors of the application domain such as the availability of labeled data, the nature of the data, types of anomalies to be detected and so on [1].

1.2 Nature of input data

The nature of the input data plays a significant role in the applicability of anomaly detection techniques. Input data can be broadly classified as sequential (e.g. voice, time-series, protein sequences, text) or non-sequential (images, record data). Additionally, each data instance can be described by one or multiple attributes (or *features*). Such instances are then referred to as *univariate* or *multivariate* respectively [31, 1].

The nature of the attributes also determines the usability of techniques. Many statistical techniques use models that require the input data to be continuous or categorical. In similar fashion, for nearest neighbor-based techniques the attributes would determine the distance measure to be used [1].

Most existing techniques deal with point data, where no relationship is assumed between the individual instances. However, in general, instances can be related to each other, as is the case in spatial, sequence or graph data [1].

Another key aspect of input data is the availability of labels, which indicate that a chosen instance is either normal or anomalous. Obtaining labeled data, which is accurate and representative, is often prohibitively expensive, as such labeling must be done manually by domain experts and requires substantial effort [1].

And while normal data is usually very well sampled, anomalous samples can be very scarce due to measurement costs or the low frequency at which anomalous events occur [2, 25]. Furthermore, it may not be possible to obtain samples of all possible anomalous behaviour, as new anomalies may arise over time.

Based on the availability of labels, anomaly detection techniques can be sorted into three categories:

- **Supervised**. Techniques trained in supervised mode assume the availability of both normal and anomalous training data. Commonly, a binary or multi-class classifier is trained, which can then provide labels for unseen data.

  Supervised methods depend on learning how to separate data classes, whereas unsupervised techniques strive to understand and explain underlying characteristics in the data. Supervised

anomaly detection methods have exhibited superior performance in comparison to their unsupervised counterparts [37]. Another advantage of supervised methods is their fast classification of unseen data, as each instance only has to be compared against a precomputed model [31].

However, despite their potentially good performance, supervised methods aren't very popular due to the lack of availability of properly labeled data instances, especially when it comes to anomalous classes, which is a common occurrence in many application domains. The performance of supervised methods is also greatly affected by class-imbalances in training data sets, when some labeled anomalous data is actually available [31].

- **Semi-supervised**. Semi-supervised anomaly detection methods assume that training data contains only normal instances. These techniques aim to learn a discriminative boundary around the normal instances, and during the testing phase, any instance that does not belong to the majority class is labeled as anomalous [31].

  Semi-supervised methods are widely applicable, as they do not rely on labeled anomalous data, and normal instances are generally fairly easy to obtain [1]. These methods also offer robustness against the emergence of new types of anomalies.

- **Unsupervised**. Techniques in this category make an implicit assumption that anomalous data is fairly scarce in comparison to normal instances in the provided data set. If this assumption doesn't hold, they suffer from a high rate of false positives [1].

  These methods aim to learn to detect anomalies solely on the intrinsic properties of the given data. Unsupervised techniques can be cost-effective, as they do not require labeled data for training. However, they are also very sensitive to noise and data corruption and generally perform worse than supervised or semi-supervised alternatives [31].

Many semi-supervised anomaly detection techniques can be easily adapted to operate in an unsupervised manner [1]. It should be noted that the line between those two approaches is very thin and a lot

of works that claim the use of unsupervised methods in fact use semi-supervised techniques, as their training sets do not contain unlabeled anomalies.

1.3 Types of anomalies

An important aspect of anomaly detection techniques is the nature of anomalies to be detected. Chandola et al. [1] provide a classification of anomalies into three main types:

- **Point anomalies**. The simplest of the three types. If an individual data instance can be considered as anomalous with respect to the rest of the data, it is a point anomaly. These often happen randomly and may have no particular interpretation [31]. Illustrated in Figure 1.1, this is the most common anomaly type and as such is the focus of the majority of research on anomaly detection. That is the case of this thesis as well.

- **Contextual anomalies**. A contextual anomaly is a data instance that can be considered as anomalous under specific circumstances, but not otherwise. These have been most commonly explored in time-series or spatial data.

- **Collective anomalies**. If a group of data instances can be considered anomalous with respect to the entire data set, such occurrence is termed as a collective anomaly. The individual instances might very well not be anomalous by themselves, but their occurrence together is. These have been explored mainly for sequence and spatial data.

While point anomalies can occur in any data set, collective anomalies can happen only in data sets in which the individual instances are related. Contextual anomalies on the other hand depend on the existence of contextual attributes in the data set [1].

1.4 Detection outputs

Another important aspect of anomaly detection techniques is the manner in which the anomalies are reported. Generally, the outputs produced by anomaly detection fall into two categories [1]:

- **Anomaly Score**. Most techniques assign each tested data instance an anomaly score, based on the degree to which that instance is considered an anomaly. Individual instances may be ranked according to this score. Domain expert can then either choose to further analyze top few ranking instances, or devise a domain-specific cut-off threshold to identify anomalies. In general, anomaly scores reveal more information than labels and the decision threshold is easier to manipulate to suit specific needs [31].

- **Label**. Instead of assigning a score open for further interpretation, some techniques outright assign a label (*normal* or *anomalous*) to each tested data instance.

1.5 Surveys of anomaly detection methods

Survey made by Chandola et al. [1] remains the most complete and cited source on anomaly detection methods. The work describes in great detail how researches have adopted concepts from various disciplines such as *statistics*, *machine learning*, *data mining*, *information theory*, *information theory* and *spectral theory* and applied them to specific anomaly detection problem formulations. It also introduces many concepts described in previous sections.

Pimentel et al. [2] built upon the previous survey and introduced a classification of anomaly detection methods into five groups — *Probabilistic novelty detection*, *Distance-based methods*, *Reconstruction-based methods*, *Domain-based methods* and *Information-theoretic methods*.

Ehret and Davy [38] expanded upon the previous works and provided another exhaustive survey of anomaly detection methods specialized in image data.

Chalapathy and Chawla [31] were the first to introduce an in-depth survey of deep learning-based anomaly detection methods, which have recently become the state-of-the-art, across a wide range of application domains. And it's these methods that are the main topic of the next chapter.

2 Deep learning methods for anomaly detection in image data

In many domains, data is growing rapidly in size and dimensions, and thus arises a need for new, more efficient and scalable methods of anomaly detection. Classical anomaly detection methods often fail in high-dimensional, data-rich scenarios due to bad scalability and the curse of dimensionality. Substantial manual feature engineering is required to make them at least somewhat effective [39].

In comparison, **deep learning** [40] presents a way to learn rich and descriptive features automatically, and has shown to be superior to classical methods in many fields [41, 42], especially in computer vision [29]. Since neural networks used in deep learning are trained, not programmed, approaches using deep learning often require less expert analysis and fine-tuning and thrive off the tremendous amount of data available. Deep learning also comes with the advantage of flexibility, as models can be re-trained using a custom data set for any use case, whereas traditional methods tend to be more domain specific. The most significant trade-off is that deep learning requires more computing resources and higher training time [43].

When considering image data, which are the main concern of this thesis, survey of deep learning-based anomaly detection methods by Chalapathy et al. [31] proposes several key advantages of deep learning over classical methods:

- As the volume of data increases, it becomes impossible for classical methods to scale, while deep learning-based approaches on the other hand generally perform even better when provided with more data.

- The performance of classical anomaly detection methods on image data is sub-optimal, as these approaches fail to capture complex structures in the data. In contrast to this, convolutional neural networks are producing state-of-the-art results in many computer vision tasks [40].

- Deep anomaly detection techniques learn hierarchical discriminative features from the data itself, and thus eliminate the need for manual feature creation by experts.

Chalapathy et al. [31] also provide a categorization of deep anomaly detection techniques. Such categorization is something that has been lacking after the recent rise of deep learning, as taxonomies provided by Chandola et al.[1] and Pimentel et al. [2] aren't entirely applicable to newly proposed methods. However, Chalapathy et al. [31] fail to mention a branch of deep learning which has also been recently exploited in anomaly detection setting. Deep distance metric learning-based anomaly detection methods have recently found considerable success [25, 44]. The theses of Alaverdyan [15] and Andrews [36] too provide a nice overview of deep learning anomaly detection and both mention the use deep distance metric learning, but in turn omit some of the methods that Chalapathy el al. [31] provide.

This chapter is about combining their knowledge and proposing an up-to-date overview of deep anomaly detection techniques used on image data. Each section describes the idea behind one deep anomaly detection technique, the most commonly employed neural network models by that technique, and some recent applications.

2.1 Reconstruction-based methods

Techniques based on input reconstruction are one of the oldest cases of using neural networks for the task of anomaly detection [1, 2, 38]. These techniques try to find lower-dimensional embeddings of the input data, where normal and anomalous instances are expected to be separated from each other. When reconstructing the input data from these embeddings, we expect to obtain the true nature of the data, without noise and uninteresting features. Reconstruction error of the tested data instance, which is computed as a difference between the original and reconstructed datum, then serves as an anomaly score [45].

Reconstructive models perform the best when trained exclusively on normal data, which is fairly easy to obtain in most anomaly detection cases, as the models fail to reconstruct previously unseen features of the anomalous instances, therefore producing higher reconstruction errors [31, 45]. This stands to a reason, as during training the low-dimensional embedding serves as an information bottleneck. Due to this bottleneck the trained model cannot reconstruct every input sample well. Instead, to minimize the overall training error, it has to learn the most common features of the normal data instances [46].

Classical anomaly detection methods employ the Principal Component Analysis to perform dimensionality reduction [45]. However, PCA can only offer linear dimensionality reduction. In contrast, the most prominent neural network model used for reconstructions, the **Autoencoder**, enables the use of both linear and non-linear transformations [31].

2.1.1 Autoencoders

First proposed in 1985 [47] and later promoted by Hinton and Salakhutdinov [48], autoencoders (AEs) represent an extremely popular unsupervised method for learning internal representation [36].

A basic autoencoder is a one-hidden-layer neural network consisting of two essential parts called *encoder* and *decoder*. The encoder is a mapping that is applied on an input vector $\mathbf{x} \in R^d$ in order to transform it to a latent representation $\mathbf{z} \in R^{d'}$ (typically $d' < d$). Usually, it is modelled with a non linearity function applied to the affine trans-

formation of the input [15]:

$$\mathbf{z} = f(W\mathbf{x} + \mathbf{b}) \tag{2.1}$$

Where f is the (non)-linearity of choice, W is a $d \times d'$ weight matrix and $\mathbf{b}$ is a bias vector. The decoder is an inverse mapping from the latent space to the original space with a smiliar configuration [15]:

$$\mathbf{x}' = f'(W'\mathbf{z} + \mathbf{b}') \tag{2.2}$$

The parameter set $\Theta = \{W, W', \mathbf{b}, \mathbf{b}'\}$ is optimized so as to minimize the reconstruction error, which is the distance between original input $\mathbf{x}$ and its reconstruction $\mathbf{x}'$. Typically, mean squared error or cross-entropy are employed as the metrics to quantify the deviation between the two [15]. The loss function across all instances of data set D composed of N samples is:

$$L(D, \Theta) = \frac{1}{N} \sum_{i=1}^{N} c(\mathbf{x}_i, \mathbf{x}'_i) \tag{2.3}$$

where $\mathbf{x}_i$ is the i-th sample, $\mathbf{x}'_i$ its reconstruction and c the chosen measure of the reconstruction error.

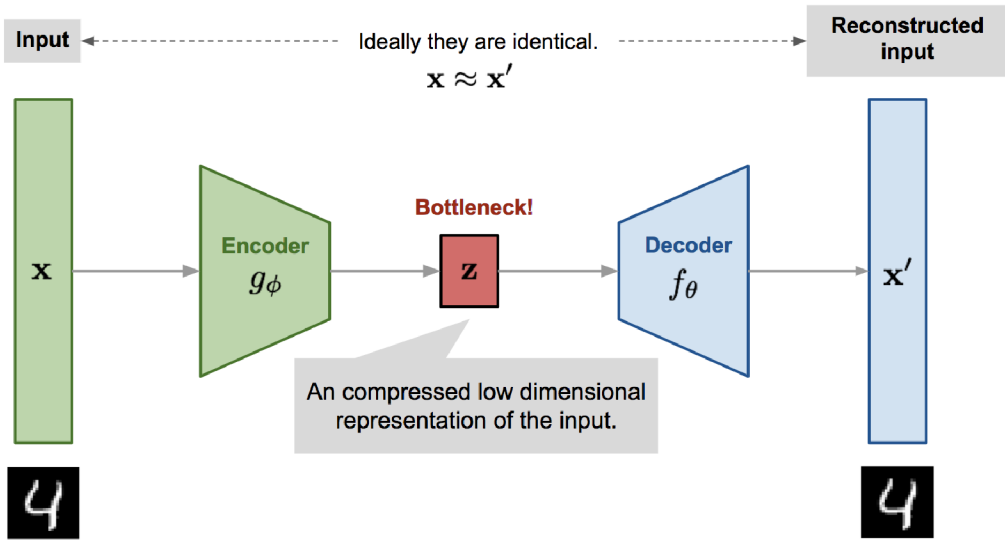


Figure 2.1: Illustration of autoencoder architecture and principles. Courtesy of [49].

Under some configuration, it is possible for the autoencoder to learn the identity mapping. Such case is not very helpful, as the network then does not learn any useful representations of the input data. Therefore, employing constraints such as $d' < d$ and non-linear activation functions in the encoder-decoder mappings (sigmoid, ReLU) has become the norm [15]. Various other methods of regularization have been used and resulted in for example sparse [50] or contractive autoencoders[45, 51].

Autoencoders can be stacked in a layer-wise manner where the latent representation of one layer serves as the input of another layer [15]:

$$\mathbf{z}^k = f(W^k \mathbf{z}^{k-1} + \mathbf{b}^k) \tag{2.4}$$

Stacked autoencoders allow to learn more complex mappings and therefore more abstract representations in the hidden layers [15].

When used for anomaly detection, autoencoders are usually trained on normal instances in a semi-supervised manner (but they can be trained in unsupervised fashion as well). After training concludes, autoencoders generally reconstruct normal instances very well, but produce significantly higher reconstruction errors on anomalous data [45]. The most common way to detect anomalies is to declare a threshold value, where every input with higher reconstruction error than the threshold is classified as anomalous. There are other ways to utilize autoencoders for anomaly detection, but these will be discussed in later sections.

2.1.2 Denoising autoencoders

Another interesting autoencoder variant is the denoising autoencoder (DAE) [52]. During training, each input sample is randomly corrupted by adding Gaussian, salt-and-pepper or zero-masking noise, and the DAE is tasked with recovering the original input [36]. To succeed at this objective, the model has to discover and capture relationships between the dimensions of the input data to infer the missing pieces. This allows the DAE to learn robust latent representations [49].

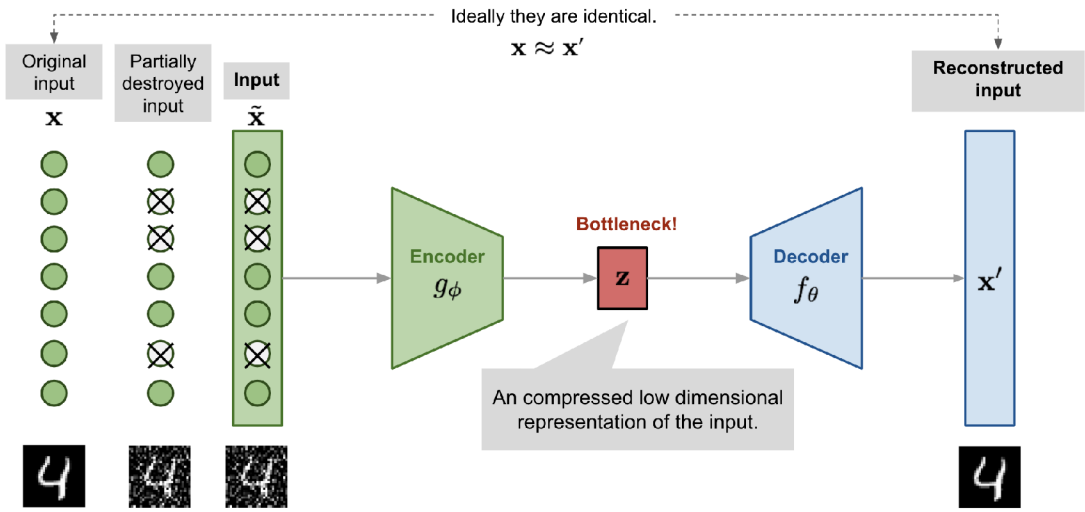


Figure 2.2: Illustration of denoising autoencoder architecture. Courtesy of [49].

2.1.3 Convolutional autoencoders

The main weakness of simple autoencoders is that they ignore the spatial relationships present in image data and as a consequence tend to learn redundant features. In contrast, convolutional neural networks make use of the convolution operator and weight sharing so as to learn features that are common in various locations of the image. Convolutional autoencoders are built upon the same idea. The input image $\mathbf{x}$ is mapped to a number of feature maps where each individual feature map $\mathbf{z}^k$ is computed as [15]:

$$\mathbf{z}^k = f(\mathbf{x} * W^k + \mathbf{b}^k) \tag{2.5}$$

where $*$ denotes the convolution operation, f is the chosen activation function, W^k the shared weight matrix and $\mathbf{b}^k$ the shared bias. The reverse mapping is done similarly.

Due to weights and biases being shared across many input locations, the amount of parameters to be trained is substantially reduced [6]. Naturally, convolutional autoencoders can also be stacked to form more complex architectures capable of capturing even more intricate patterns [15]. When mentioning autoencoders in later chapters, espe-

cially in Chapters 4 and 5, it is their convolutional variant that is being mentioned.

2.1.4 Recent applications

Some recent anomaly detection applications of different autoencoder architectures include the research of Sato et al. [53] concerning emergency CTs, early detection of respiratory disease in pigs by Cowton et al. [54], the detection of fake twitter followers by using a denoising autoencoder [55], the monitoring of power equipment based on stacked denoising autoencoders [56], the detection of anomalous business processes [57], gas turbine engine gas path anomaly detection [58], the detection of pitting in gears using deep sparse auteoncoders [59], anomalous power consumption detection [60], industrial surface inspection [61] or DDoS attack detection [62].

Autoencoder architectures for anomaly detection are still extensively studied as shown by the survey of Chalapathy et al. [31] or the work of Ribeiro et al. [63]. Recent research of Munawar et al. [64] identifies possible enhancement of the training phase of autoencoder architectures. They introduce a concept of *negative learning*, where at some of the steps during the training of the autoencoder model, real or synthesised anomalous input is also being used, and the model is penalised for learning patterns that occur in anomalous data.

2.2 Distance metric-based methods

Several computer vision tasks such as verification, matching or retrieval share a common need of being able to measure the similarity between pairs of images. Deep distance metric learning (DDML) uses neural networks to directly learn such similarity measure, instead of creating it as a by-product of solving e.g. a classification task [25]. The general goal is to learn an end-to-end parameterized function f_θ that maps data samples from a high-dimensional input space to a low-dimensional feature space, where samples of one class have lower distances to each other than to samples of different classes [44]. By employing specialised loss functions, the neural network models learn to discern patterns unique to each class of the input data.

Significant advantage of DDML is that the models can generalise to classes unobserved during training, and are especially useful in settings where the number of classes is potentially infinite and simple classification is not feasible anymore [25]. For the task of anomaly detection, this feature of DDML is key in domains where the number of types of anomalies is large, or where new anomalies tend to appear as time passes. And while some samples of anomalous data are required during the training phase, popular DDML models have shown to be capable of being efficiently trained on very little data, even despite class imbalances [65, 66, 67].

When applying DDML models on the task of anomaly detection, the detection itself on the extracted features requires additional effort. Most commonly, the extracted features serve as an input to a classifier, such as one-class SVM or k-NN [15, 68]. Other approach is to use a prototype of normal data and calculate the difference between the tested sample's extracted features and those of the prototype. A threshold can then be employed to decide whether the tested sample is different enough to be classified as an anomaly [25].

Two DDML models rose to prominence in the field of anomaly detection. They are the **Siamese network** and the **Triplet network**. Short description of both follows.

2.2.1 Siamese networks

Siamese neural networks were first introduced by Bromley et al. [69] for the task of signature verification. A siamese network consists of two sub-networks (sometimes called *branches*) with identical architecture, which are constrained to share the same weights, and a cost module. It receives a pair of samples as an input and propagates each of them through one of the sub-networks. Each sub-network represents function G_W parameterized with W and outputs a feature vector. Both feature vectors are finally passed to a cost module which outputs their difference [15].

Weight sharing guarantees that two extremely similar images could not be possibly mapped to very different locations in the feature space by their respective sub-networks, because each sub-network computes identical function. This also makes the network symmetric [66].

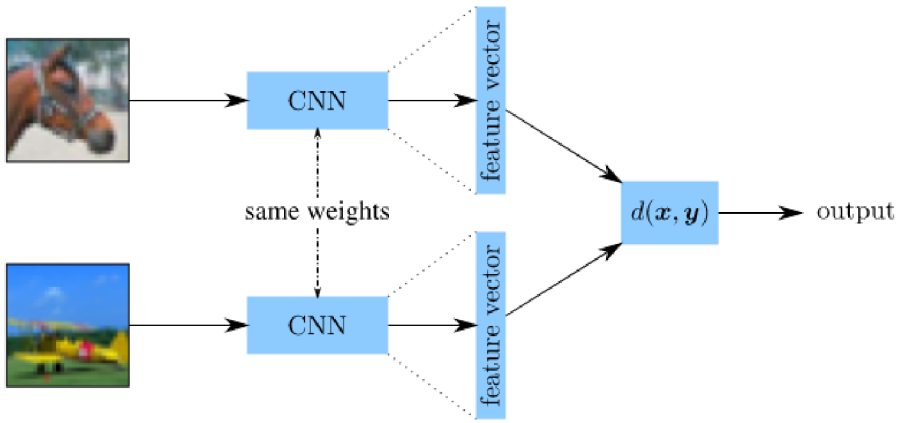


Figure 2.3: Illustration of siamese network architecture. Courtesy of [70].

With a well-chosen loss function, the model learns to map similar inputs close to each other in the feature space with respect to some measure and pulls apart the dissimilar ones.

Given a data set D consisting of pairs of input data and their labels $D = \{(x_{i1}, x_{i2}, l_i)\}$ where l_i is a label equal to 0 when the pair i is composed of similar data and 1 otherwise, the general form of the loss function of such network is [15]:

$$L(D; W) = \frac{1}{N} \sum_{i=1}^{N} [(1 - l_i) \cdot L_S(D_W^i) + l_i \cdot L_D(D_W^i)] \tag{2.6}$$

where N is the number of samples, L_S is the loss function for similar input samples and L_D for dissimilar, D_W is a distance function for calculating the similarity between the feature vectors and D_W^i designates its value for the i-th pair. The most commonly used distance function is the Euclidean distance:

$$D_W^i = \|G_W(\mathbf{x}_{i1}) - G_W(\mathbf{x}_{i2})\|^2 \tag{2.7}$$

Typical example and the most often used loss function when training siamese networks is the **Contrastive loss** introduced by Hadsell et al. [71]:

$$L(D;W) = \frac{1}{N}\sum_{i=1}^{N}[(1-l_i)\cdot\frac{1}{2}(D_W^i)^2 + l_i\cdot\frac{1}{2}max\{0,(m-D_W^i)\}^2] \tag{2.8}$$

where D_W^i the aforementioned Euclidean distance and $m > 0$ a margin parameter. The margin m determines a radius around the output of each sub-network and a dissimilar input pair only contributes to the loss function if the inputs are mapped within the distance m of each other [36]. This forces the model to learn to separate the input classes in the feature space by at least the given margin.

Another possibility is to minimize the cross-entropy between the similarity label ($\{0,1\}$) and the softmax activation computed on the outputs of the two sub-networks, as shown by Han et al. [72]:

$$L(D;W) = -\frac{1}{N}\sum_{i=1}^{N}[l_i\cdot log(\hat{l}_i) + (1-l_i)\cdot(1-log(\hat{l}_i))] \tag{2.9}$$

$$\hat{l}_i = \frac{e^{G_W(\mathbf{x}_{i2})}}{e^{G_W(\mathbf{x}_{i1})} + e^{G_W(\mathbf{x}_{i2})}} \tag{2.10}$$

2.2.2 Triplet networks

Wang et al. [73] first employed a deep learning approach that extended pairwise constraints into triplets. Triplet networks consist of three architecture-wise identical sub-networks that share weights followed by a cost module. Siamese and Triplet networks share the same underlying principle, with Triplet networks differing mainly in the input and corresponding loss function [15].

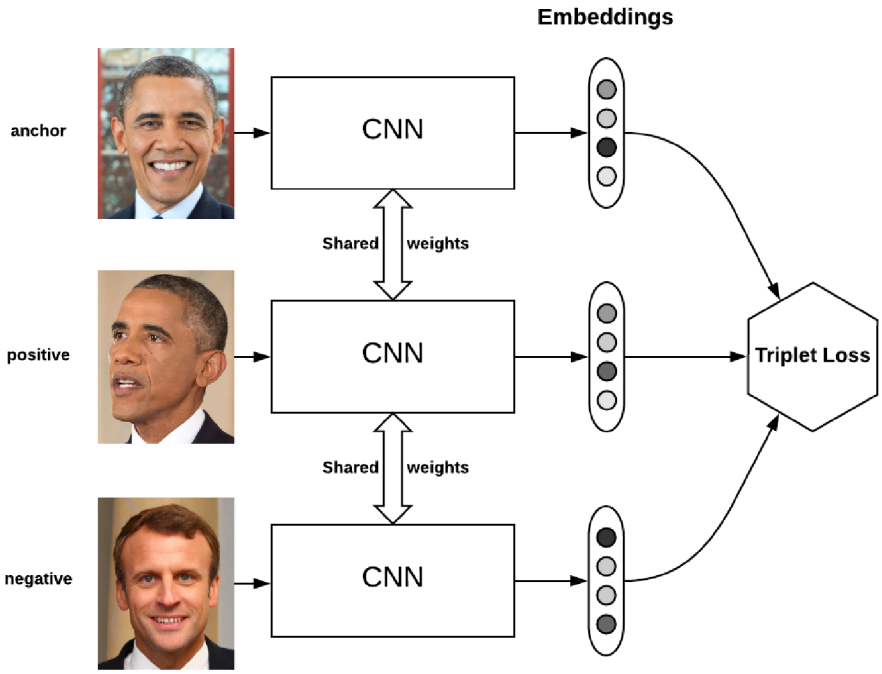


Figure 2.4: Illustration of Triplet network architecture. Courtesy of [74].

The input to this network are triplets $D = \{(a_i, p_i, n_i)\}$ consisting of *anchor* samples, *positive* samples of the same class as the *anchor* samples and *negative* samples of a different class from the *anchor* samples. The learnt function G_W should output a higher similarity score when comparing anchor and a positive sample than when comparing anchor with a negative sample. To this end, the **Triplet loss** is used, which is designed to bring closer the anchors and their positives in the feature space and to pull apart anchors and their negatives [36]:

$$L(D;W) = \frac{1}{N}\sum_{i=1}^{N}[max\{(D_W(a_i, p_i) - D_W(a_i, n_i) + \alpha), 0\}] \quad (2.11)$$

where a_i, p_i, n_i are the i-th anchor, positive and negative samples respectively, $\alpha > 0$ is a margin parameter and $D_W(x_1, x_2)$ is a distance function computing the Euclidean distance between the feature vectors of inputs x_1, x_2. The role of margin α is similar to the one used in

contrastive loss equation 2.8. Here, it pushes the difference between same-class samples to 0, while enforcing $D_W(a, n) > D_W(a, p) + \alpha$.

Several studies have shown the Triplet networks to be more effective than their Siamese counterparts [75, 76, 77]. To even further improve the performance and the convergence speed, Schroff et al. [75] propose **Triplet mining**. Triplets can be divided into three categories:

- **Easy triplets**. These triplets have loss of 0, because $D_W(a, p) + \alpha < D_W(a, n)$

- **Hard triplets**. Triplets where the negative is closer to the anchor than the positive, i.e. $D_W(a, p) > D_W(a, n)$

- **Semi-hard triplets**. Triplets where the negative is not closer to the anchor than the positive, but which still generate positive loss: $D_W(a, p) < D_W(a, n) < D_W(a, p) + \alpha$

Oftentimes, many triplets in a data set fall into the easy category. These triplets do not contribute to the training and result in slower convergence. Triplet mining is about improving the convergence through selecting only hard or semi-hard triplets. This can be done in an offline fashion, where for example at the beginning of each training epoch, the feature vectors of each triplets are computed using the current model, and only appropriate triplets are then used for the training epoch. More efficient approach is to mine triplets in an online fashion, where valid useful triplets are constructed on the fly in a batch-wise manner [74, 75].

While online triplet mining might be more efficient, it requires large batch sizes, which might not be usable in some scenarios, for example when dealing with large input samples. Selecting the hardest triplets can in practice lead to a collapsed model, so mining semi-hard triplets is the recommended approach [75].

2.2.3 Recent applications

Anomaly detection using DDML models is currently a popular anomaly detection research topic, especially due to the ability of the models to work very well on a limited amount of training data [44, 78].

Siamese and Triplet networks have recently found their anomaly detection use in several domains, including monitoring of robot behaviour [79], video surveillance [80], video game bug finding [81], industrial surface inspection [25, 82], machine condition monitoring [83], intrusion detection [84] or railway track anomaly detection [65].

2.3 Hybrid models

In the past, there has been a substantial effort to utilize traditional machine learning techniques for the task of anomaly detection, such as principal component analysis, isolation forests, Gaussian mixture model or k-NN. The most popular of them is the **one-class Support Vector Machine** (oc-SVM), which has produced great results when tasked with unsupervised anomaly detection and has shown to be insensitive to noise and outliers in the training data [85].

However, the performance of all these methods is highly susceptible to the growing dimensionality and complexity of the data, coupled with the training speed also being heavily affected by the size of the data set. As a result, these traditional methods aren't usable in big data and high-dimensional anomaly detection applications [85].

In comparison, deep learning [40] presents a way to learn relevant and robust features (thus performing a significant dimensionality reduction), and has found great success especially in computer vision tasks [29].

Hybrid anomaly detection methods use deep neural network models (mostly autoencoders, but distance metric models such as Siamese networks have also been used [15]) as robust feature extractors, whose output then serves as the input to classical anomaly detection methods, most prominently the oc-SVM [31, 86].

Such combination works well when unsupervised training is the only option, as autoencoders do not require labels to learn to extract meaningful features and oc-SVM is, as mentioned before, insensitive to outliers in the given data. Another advantage of hybrid models is significantly reduced anomaly detection time [87].

Combining autoencoders with oc-SVM also amends one of the possible shortcomings of using autoencoders for anomaly detection by themselves, which is the fact that the reconstruction error does

not take into account the relationships between the extracted features. For example, some anomalous samples might not produce noticeable reconstruction errors, but their feature vectors would be highly abnormal [86]. This is often the case in image data, where spatial relations have an important role.

Another possibility is to use transfer learning [14]. Transfer learning is an essential tool in machine learning for tackling a lack of training data. It aims to transfer learned knowledge from a source domain to a target domain by relaxing the assumption that training and future data must be in the same feature space and have the same distribution [31]. For hybrid anomaly detection models this would mean replacing the autoencoder as the feature extractor by a different, already trained (albeit on a different input data of the same type), deep neural network extractor. While employing transfer learning for the task of anomaly detection has led to some promising results [88, 89], research by Andrews et al. [90] shows that learning the representations from scratch, on a moderately-sized data set, for a specific task of anomaly detection still performs better. As such, the degree of usability of transfer learning for anomaly detection remains an open research question [31].

One notable shortcoming of hybrid models is that they are theoretically sub-optimal, because the representations learned within the hidden layers of the extractor network are optimised by generic loss functions, instead of a customized objective for anomaly detection. Hence the learned representations aren't the best they can be in terms of differentiating normal samples from anomalous [31]. Nonetheless, hybrid models still report state-of-the-art performance [86, 87, 91].

To better understand how the anomaly detection itself takes place, a short description of oc-SVM follows.

2.3.1 One-class SVM

The one-class SVM, introduced by Schölkopf et al. [92], is a special case of the classical SVM, which aims to find a hyperplane that would separate all of the given data from the origin, which serves as the sole *other* class representative [85].

Since most data sets aren't linearly separable in the original data space, the inputs are first mapped into a higher-dimensional space us-

ing a mapping $\phi(\mathbf{x})$ with a corresponding kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ where $\langle \cdot, \cdot \rangle$ denotes the inner product. Although different kernels may be chosen, polynomial and sigmoid kernels have been shown to systematically fail in the scope of anomaly detection [93], and as such the most commonly and successfully used kernel is the radial basis function (RBF) kernel due to its locality preserving properties [15]:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}) \tag{2.12}$$

where σ is the kernel bandwidth.

Let X be a set of n observations $X = \{\mathbf{x}_i\}_{i=1,\dots,N}$ where $\mathbf{x}_i \in \mathcal{X}$ and ϕ be a feature map $\mathcal{X} \to \mathcal{F}$ from the original space to a dot product space $\mathcal{F}$, corresponding to some kernel K as mentioned above. The oc-SVM algorithm seeks to project the input data points into a new feature space with the use of K and to separate them from the origin with a maximum margin. To that end, it solves the following quadratic optimisation function [15]:

$$\min_{\mathbf{w},\rho,\xi_i} \ \frac{1}{2}\|\mathbf{w}\|^2 - \rho + \frac{1}{vN}\sum_{i=1}^{N} \xi_i \tag{2.13}$$

subject to:

$$\begin{aligned} \mathbf{w} \cdot \phi(\mathbf{x}_i) &\geq \rho - \xi_i && \forall i = 1, \dots, N \\ \xi_i &\geq 0 && \forall i = 1, \dots, N \end{aligned}$$

where N is the number of training samples, $\mathbf{x}_i$ is the i-th sample from the training set $\mathcal{X}$, $\mathbf{w}$ and ρ define the separating hyperplane, ξ_i are slack variables relaxing the inequality constraints and $v \in (0, 1]$ is a parameter that sets an upper bound on the fraction of training samples that can be classified as outliers and a lower bound on the fraction of training samples that can be used as support-vectors [86].

When the optimal solution $\mathbf{w}^*, \rho^*$ is found, the decision for a sample $\mathbf{x}$ depends on the side of the hyperplane it falls in.

Let $g(\mathbf{x}) = \mathbf{w}^* \cdot \phi(\mathbf{x}) - \rho^*$, the decision function can then be expressed as [85]:

$$f(\mathbf{x}) = \text{sign}(g(\mathbf{x})) = \begin{cases} +1 & \text{if } g(\mathbf{x}) \geq 0 \\ -1 & \text{if } g(\mathbf{x}) < 0 \end{cases} \tag{2.14}$$

Trained oc-SVM offers a fast and clearly interpretable anomaly prediction, where $+1$ is assigned to normal samples and -1 to anomalous ones.

The parameter v allows the oc-SVM to be trained in unsupervised manner, where a small portion of the training data set can consist of anomalous samples, as such anomalies will most likely be found out and correctly separated by the learned hyperplane. In semi-supervised manner, with the training data set containing only normal samples, v might cause that some samples will be incorrectly classified as false anomalies, as the parameter cannot be set to 0. In both cases, the parameter v might need tuning for the oc-SVM to produce the best results. Another possibility, when some sort of performance-testing data set isn't available, is to use an ensemble of oc-SVM classifiers, each with a different v value, and use the majority vote to classify input samples [86].

The optimization function of oc-SVM is usually solved as a convex optimisation problem in the dual space, with the use of Lagrangian multipliers to reduce the complexity [85, 92]. Further description of the process is beyond the necessary scope of this section.

One closely related technique to oc-SVM, which warrants a mention, is the **Support Vector Data Description** (SVDD) [94]. SVDD operates on a similar principle to oc-SVM, the difference being that this technique tries to find the smallest possible *hypersphere* with center $\mathbf{c} \in \mathcal{F}$ and radius $R > 0$ that encloses the majority of the data in the feature space $\mathcal{F}$. Its primal problem is formulated as [39]:

$$\min_{R,\mathbf{c},\xi_i} \; R^2 + \frac{1}{vN} \sum_{i=1}^{N} \xi_i \tag{2.15}$$

subject to:

$$\begin{aligned} \|\phi(\mathbf{x}_i) - \mathbf{c}\|_{\mathcal{F}}^2 &\leq R^2 + \xi_i && \forall i = 1, \ldots, N \\ \xi_i &\geq 0 && \forall i = 1, \ldots, N \end{aligned}$$

where again, slack variables ξ_i allow a soft boundary and parameter $v \in (0, 1]$ serves the same purpose as in oc-SVM. Points which fall outside the sphere, i.e. $\|\phi(\mathbf{x}_i) - \mathbf{c}\|_{\mathcal{F}}^2 > R^2$, are considered anomalous [39]. As with oc-SVM, the solution is generally found by solving the dual problem.

Finally, both techniques have been proven to be equivalent for certain types of kernels, such as the most widely used RBF kernel [95].

2.3.2 Recent applications

Despite being a relatively new approach, various hybrid models have already found success in domains such as medical [15, 96], video surveillance [97], depression detection [98], disaster area detection [99], diagnosis of bearing faults [100], network intrusion detection [84], DDoS attack detection [101] and general network traffic anomaly detection [102, 103].

While being theoretically sub-optimal, these methods are more often than not still an improvement over anomaly detection using their individual parts in most domains, and are also applicable in domains where the individual techniques would fail.

2.4 Generative model-using methods

Another field of machine learning, that has very recently risen in popularity thanks to application of the advancements of deep learning, is generative modeling [104]. Projects such as StyleGAN [105] or more recently GPT-3 [106] have garnered much attention. Naturally, with the rise of deep generative modelling came an interest in employing such models for the task of anomaly detection.

Let $p(\mathbf{x})$ be the distribution of our data. Generative modelling aims to capture this distribution using probabilistic models. However, the exact modelling of this probability density function is unfortunately computationally intractable in all but the most trivial of models. Deep generative models hence use different ways of how to approximate $p(\mathbf{x})$ [107]. The principles of the most commonly used generative models are described in the following subsections.

Generative models are similar to reconstructive models and can be thought of as a generalization of them. The main difference being that generative models are *stochastic* and learn the distribution of some data, while reconstructive models are discriminative and learn a *deterministic* mapping function [45]. Learning the distribution allows generative models to produce convincing not-yet-seen examples by

sampling the distribution [108], while reconstructive models lack this ability.

Another difference is that the latent space is continuous in the case of generative models. The consequence of which is that interpolations in the latent space produce smooth and plausible semantic variations, and certain directions in this space correspond to a particular semantic attribute along which the data distribution varies [107, 109].

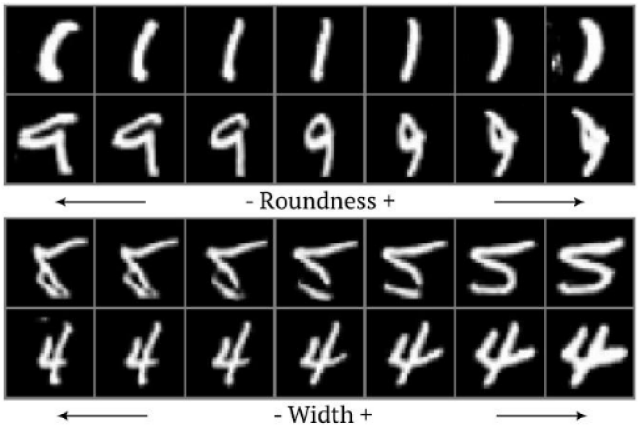


Figure 2.5: An illustration of how trajectories in the latent space correspond to semantic variations. Courtesy of [109].

For the task of anomaly detection, generative models are trained in semi-supervised or unsupervised fashion. The goal is to learn the distribution $p(\mathbf{x})$ of normal data. The idea behind detecting anomalies is the same as in the case of reconstructive models, which is that the trained model will fail to reconstruct anomalous samples, as they come from a different distribution. To this end, reconstruction error is once again employed, but for many generative architectures it serves as only one of the parts of the custom anomaly score functions [6, 110].

When it comes to modelling complex distributions such as image data, the two most efficient and powerful approaches are **Variational Autoencoders** (VAE) and **Generative Adversarial Networks** (GAN) [31]. Their description follows.

2.4.1 Variational autoencoders

The older of the two approaches, the variational autoencoder was introduced by Kingma and Welling [111]. It is a directed probabilistic graphical model of two variables, an observed $\mathbf{x}$ and an unobserved $\mathbf{z}$, where the distribution of given data $p(\mathbf{x})$ is modeled using the latent $\mathbf{z}$ [108]. The variable $\mathbf{z}$ is generated from some prior distribution $p_\theta(\mathbf{z})$ (most commonly a multivariate Gaussian) while $\mathbf{x}$ is generated from some conditional $p_\theta(\mathbf{x}|\mathbf{z})$. As the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, but necessary for training of the model, an approximation $q_\phi(\mathbf{z}|\mathbf{x})$ is introduced [15].

The objective of the graphical model is to maximimze the log likelihood of a given dataset $X = (\{\mathbf{x}_1, \ldots, \mathbf{x}_N\})$ composed of N samples, i.e. $\sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i)$, where $\log p_\theta(\mathbf{x}_i)$ can be written as [15]:

$$\begin{aligned} \log p_\theta(\mathbf{x}_i) &= D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \| p_\theta(\mathbf{z}|\mathbf{x}_i)) \\ &+ \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \| p_\theta(\mathbf{z}))}_{\mathcal{L}(\theta,\phi;\mathbf{x}_i)} \end{aligned} \tag{2.16}$$

where D_{KL} is the Kullback-Leibler divergence and $p_\theta(\mathbf{x}_i|\mathbf{z})$ the likelihood of instance $\mathbf{x}_i$ given the latent variable $\mathbf{z}$. Due to the intractability of the first term, the model instead optimizes $\mathcal{L}(\theta, \phi; \mathbf{x}_i)$, which is the lower bound of the overall likelihood, as KL-divergence is non-negative [15].

The second term of $\mathcal{L}(\theta, \phi; \mathbf{x}_i)$ is the KL-divergence between the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior of the latent variable $\mathbf{z}$. This term forces the approximate posterior to be similar to the prior distribution, working as a regularization term [45]. The first term can be interpreted as the reconstruction of $\mathbf{x}$ through the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ [15]. For Gaussian distributions, this becomes the mean squared error [108].

In practice, the reconstruction loss is usually weighted with some factor α_{rl}, that ensures that the reconstruction loss is well balanced with the KL-divergence loss. Otherwise, the reconstruction quality or generative capability might suffer. As such, the reconstruction loss factor α_{rl} becomes another hyperparameter to tune during training [104].

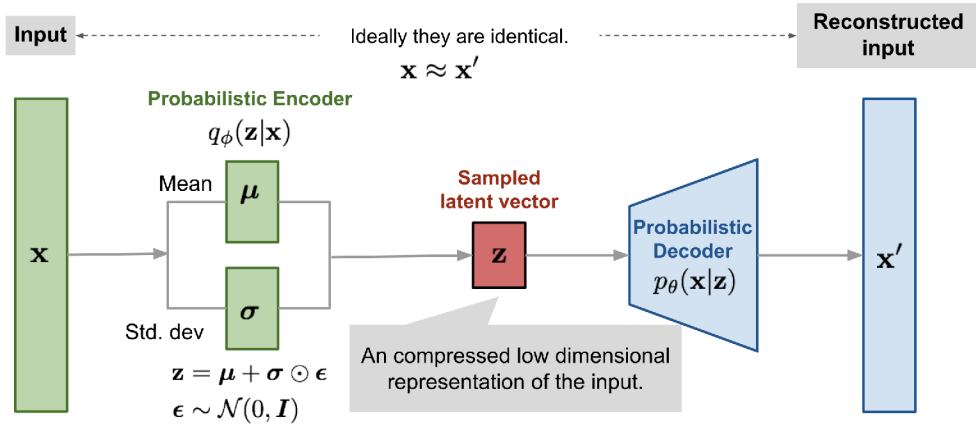


Figure 2.6: Illustration of variational autoencoder model with the multivariate Gaussian assumption. Courtesy of [49].

In variational autoencoders, the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is modeled with the use of a neural network. Figure 2.6 illustrates the relation of VAEs to classical autoencoders, where the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is the encoder part and the directed probabilistic graphical model $p_\theta(\mathbf{x}|\mathbf{z})$ is the decoder part. VAE aspires to model the parameters of the distributions rather than the distributions itself. That is, the encoder function $f(\mathbf{x}, \phi)$ outputs the parameters of the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and to obtain the actual value of the latent variable $\mathbf{z}$, sampling from $q_\phi(\mathbf{z}; f(\mathbf{x}, \phi))$ is required. To get the reconstruction $\mathbf{x}'$ given the sample $\mathbf{z}$, the parameters of $p_\theta(\mathbf{x}|\mathbf{z})$ are obtained by the decoder function $g(\mathbf{z}, \theta)$ and the reconstruction $\mathbf{x}'$ is then sampled from $p_\theta(\mathbf{x}; g(\mathbf{z}, \theta))$. Thus, the encoder and decoder of VAE can be called probabilistic [45]. One should note however, that in practice many VAE architectures use decoder network which outright outputs the reconstruction $\mathbf{x}'$, instead of the learned parameters of the training distribution [108, 110]. This approach is equivalent to modelling the $p_\theta(\mathbf{x}|\mathbf{z})$ as Gaussian with identity covariance [108].

The expectation term in the loss function invokes generating samples from $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. As sampling is a stochastic process, the gradient cannot be backpropagated during training. The solution to making the network trainable is the reparameterization trick. It is often possible

to express the random variable $\mathbf{z}$ as a deterministic variable using a transformation function $h_\phi(\mathbf{x}, \boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ is an independent random noise variable drawn from a standard normal distribution, and the transformation function h_ϕ converts ϵ to $\mathbf{z}$ [49].

The produced parameter estimates from the encoder network are used together with the introduced noise variable to sample variable $\mathbf{z}$. Common setting is to consider that the variable $\mathbf{z}$ is drawn from a multivariate Gaussian with a diagonal covariance structure and have the encoder network output its mean $\boldsymbol{\mu}$ and variance σ which are then used to draw sample $\mathbf{z}$ [15]:

$$\mathbf{z} = \boldsymbol{\mu} + \sigma \cdot \boldsymbol{\epsilon} \tag{2.17}$$

where the right hand side is the transformation function $h_\phi(\mathbf{x}, \boldsymbol{\epsilon})$.

When tasked with anomaly detection, it is possible to use the simple MSE of the input and its reconstruction as the anomaly score, coupled with a decision threshold, just like when using the classical autoencoders. As VAE learns the distribution of normal data, it should fail to reconstruct samples coming from the anomalous distribution and therefore produce high reconstruction errors on such samples [110].

Another possibility is to use the reconstruction probability introduced by [45]. In this setting, L samples are drawn from the trained probabilistic encoder when testing the input $\mathbf{x}$. For each drawn sample, the probabilistic decoder outputs the mean and variance parameters (the decoder produces parameters of the training distribution in this case, not the input reconstruction $\mathbf{x}'$). Using these parameters, the probability of the input sample $\mathbf{x}$ originating from the training distribution is calculated. The average probability of all drawn samples then serves as an anomaly score and is called the reconstruction probability [6, 45]:

$$p_{\text{recon}}(\mathbf{x}_i) = \frac{1}{L} \sum_{i=1}^{L} p_\theta(\mathbf{x}_i | \boldsymbol{\mu}_{\mathbf{x}'_{i,l}}, \sigma_{\mathbf{x}'_{i,l}}) \tag{2.18}$$

Once again, a threshold value is needed to classify the tested sample as anomalous or not.

2.4.2 Generative adversarial networks

Generative adversarial networks (GAN) were first introduced by Goodfellow et al. [112] and have since seen a variety of extensions. A GAN consists of two adversarially trained neural networks, a generator G and a discriminator D. The generator's objective is to produce data samples as realistic as the authentic ones given in a training set. The discriminator in turn learns to discriminate between real samples and fake samples produced by the generator [110].

To learn a generative distribution p_g over the training data $\mathbf{x}$, the generator builds a mapping from a prior noise distribution p_z (e.g. a normal or uniform distribution) to the training data space as $G(\mathbf{z})$. This is somewhat similar to the decoder part of a VAE. The discriminator outputs a single scalar value representing the probability that the sample $\mathbf{x}$ came from the real data distribution p_{data}, rather than from the generative p_g. The discriminator function is denoted as $D(\mathbf{x})$ [110, 113].

Training a GAN implies improving the generator, so that the discriminator is not successful at distinguishing the generated fake samples from the real training data, yet at the same time improving the discriminator's performance. This corresponds to a two-player minimax game with a unique solution G^* capturing perfectly the underlying real distribution p_{data} and D^* equal to $\frac{1}{2}$ everywhere. The objective of such game therefore is [15]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1 - D(G(\mathbf{z})))] \tag{2.19}$$

Goodfellow et al. [112] showed that minimizing the above-mentioned value function leads to minimizing the Jensen-Shannon divergence between the real distribution p_{data} and the modelled p_g [107].

Goodfellow et al. [112] also propose that the actual training is to be done in an alternating fashion, switching between updating the generator and the discriminator. After a suitable number of epochs, the generator and the discriminator will have found an equilibrium that allows the generator to learn meaningful information from the discriminator and the quality of the generated data will start to improve [104]. And while this optimisation is not guaranteed to converge,

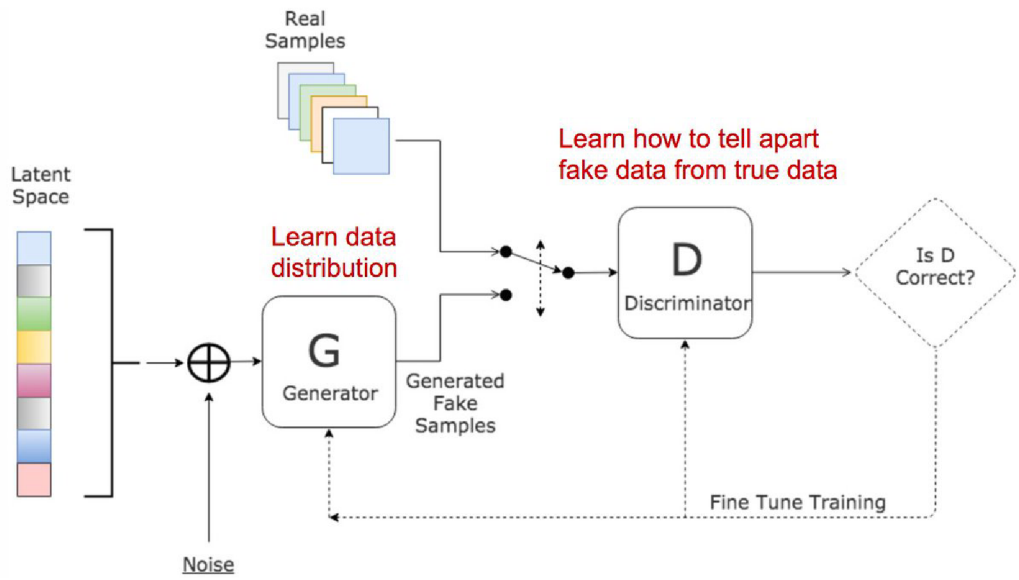


Figure 2.7: Illustration of GAN architecture. Courtesy of [114].

empirically it can work well if the discriminator and generator are well balanced [107].

While GANs enjoy success in generating realistic data, especially images, they are famously difficult to train due to their instability [15, 110]. The book on generative deep learning by David Foster [104] provides a nice overview of some of the difficulties one has to face when training GANs:

- **Oscillating loss**. Rather than exhibiting long-term stability, the loss of the discriminator and the generator can oscillate wildly. Typically, some small oscillation of the loss between batches is fine. However, in the long-term the loss should stabilize, or should gradually increase or decrease. Erratic fluctuation is a sign of a failing model.

- **Mode collapse**. Mode collapse happens when the generator finds a small number of samples (often even a single one) that fool the discriminator and therefore isn't capable of producing any other samples than this small set.

- **Uninformative loss**. It would be natural to think that the smaller the loss of the generator, the better the quality of the data produced. However, the generator is only graded against the current discriminator, which is constantly improving, and thus the loss function cannot be compared across different points of the training process. Indeed, the generator loss can actually steadily increase, even while the generator is producing higher quality images. This lack of correlation makes monitoring the training process difficult.

- **Hyperparameter tuning**. Even with simple GANs, there is a large amount of hyperparameters to tune. The overall architecture plays a significant role in the training process, as well as the parameters that govern the batch normalization, dropout, learning rate, activation layers, batch size, convolutional filters and the size of the latent space. GANs are especially sensitive to very slight changes in all of these parameters, and finding a set that actually works is often a case of educated trial and error.

2.4.3 Wasserstein GAN

Further works proposed a number of different modifications of the basic GAN in order to improve the training stability and to introduce a more informative loss [15]. One such effort resulted in the **Wasserstein** GAN model [115].

With a relatively few changes, the authors were able to show how to train GANs that have the following properties [104]:

- A meaningful loss metric that correlates with the generator's convergence and produced sample quality.

- Improved stability of the optimization process.

The paper introduces a new loss function for both the generator and the discriminator network. Usually, the GAN model is trained using the binary cross-entropy loss and using $y_i = 1$ as a label for the real data and $y_i = 0$ for the generated data, when training the discriminator network. The authors propose the *Wasserstein* loss, which requires the use of $y_i = 1$ and $y_i = -1$ as labels for the real or generated samples

respectively. The sigmoid activation function from the final layer of the discriminator network is also removed, therefore the predictions p_i are no longer constrained to the range $[0, 1]$, but instead can be any number from $[-\infty, \infty]$. In this setting, the discriminator is usually referred to as the *critic*. The Wasserstein loss is then defined as follows [104]:

$$L_W = -\frac{1}{N}\sum_{i=1}^{N}(y_i p_i) \tag{2.20}$$

where N is the number of samples, y_i the label for the i-th sample and p_i the prediction for i-th sample.

To train the critic, its loss is calculated by comparing predictions for real samples $p_i = D(\mathbf{x}_i)$ to the label $y_i = 1$ and predictions for generated samples $p_i = D(G(\mathbf{z}_i))$ to the label $y_i = -1$. The critic then minimizes the following function [104]:

$$\min_D V(D, G) = -\left(\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z}[D(G(\mathbf{z}))]\right) \tag{2.21}$$

Intuitively, the WGAN critic tries to maximize the difference between its predictions for real samples and generated samples, with the real samples scoring higher.

To train the generator, its loss is calculated by comparing predictions for generated samples $p_i = D(G(\mathbf{z}_i))$ to the label $y_i = 1$. The generator then minimizes the following function [104]:

$$\min_G V(D, G) = -\left(\mathbb{E}_{\mathbf{z}\sim p_z}[D(G(\mathbf{z}))]\right) \tag{2.22}$$

As the critic output can be in the range $[-\infty, \infty]$, the Wasserstein loss can be enormous. For the loss to work well, its authors place additional constraints on it. Specifically, it is required that the critic is a *1-Lipschitz continuous function*. To enforce this constraint, the authors resort to clipping the weights of the critic network to lie in a small range, for example $[-0.01, 0.01]$ [104].

When using the Wasserstein loss function, the critic should be trained to convergence, to ensure that the gradients for the generator update are accurate. This is in a stark contrast to the original GAN, where it is important not to let the discriminator get too strong, as that introduces the problem of vanishing gradients. Therefore, using

this loss removes one of the many difficulties of training the GAN model, which is how to balance the training of the discriminator and the generator. Typically, the critic is trained several times between the generator updates to ensure it is close to convergence [104].

As a final remark on WGANs, the authors themselves proclaim that the weight clipping is a terrible way to enforce the critic constraints and encourage others to devise better solutions. They also note that momentum-based optimizers such as Adam lead to instability when used with high learning rates, and advise to use the RMSProp instead [115].

2.4.4 anoGAN

While GANs are powerful generative models, their usage for the task of anomaly detection is not intuitive. When the adversarial training is complete, the generator has learned the mapping $G(\mathbf{z})$ from latent space representations $\mathbf{z}$ to realistic data samples $\mathbf{x}$. However, GANs do not yield the reverse mapping $\mu(\mathbf{x})$ from the training data manifold $\mathcal{X}$ to the latent space $\mathcal{Z}$, therefore they do not allow to evaluate a likelihood [6, 113].

The first successful attempt to use a GAN model for anomaly detection was the work of Schlegl et al. [113], for which they coined the name **anoGAN**. Their approach stems from the observation that the latent space $\mathcal{Z}$ has smooth transitions, so sampling two points close to each other in the latent space generates two similar samples from the real data space.

Given a query sample $\mathbf{x}$, their aim is to find a point $\mathbf{z}$ in the latent space, which corresponds to a generated sample $G(\mathbf{z})$ that is the most similar to the query sample $\mathbf{x}$ and is located on the manifold $\mathcal{X}$. When the model is trained in a semi-supervised fashion, the generator learns to generate only normal samples. When an anomalous sample is encoded to the latent space and then reconstructed, the reconstruction will be non-anomalous and hence the difference between the original sample and its reconstruction will highlight the anomalies [116].

The authors have defined the mapping of input samples to the latent space as an iterative process. This process begins by sampling a random $\mathbf{z}_1$ from the latent space and feeding it to the trained generator, providing a reconstructed sample $G(\mathbf{z}_1)$. Based on this reconstructed

sample, a loss function is utilized, which provides gradients for the update of the coefficients of $\mathbf{z}_1$, resulting in updated position in the latent space $\mathbf{z}_2$. To find the most similar sample $G(\mathbf{z}_\Gamma)$, the location of $\mathbf{z}$ in the latent space is optimized iteratively in up to $\gamma = 1, \ldots, \Gamma$ backpropagation steps [113].

The utilized loss function defined by the authors comprises of two components, the *residual* loss and the *discrimination* loss. The residual loss enforces the similarity between the query sample $\mathbf{x}$ and its reconstruction $G(\mathbf{z}_\gamma)$ and is defined as [113]:

$$L_R(\mathbf{z}_\gamma) = \sum |\mathbf{x} - G(\mathbf{z}_\gamma)| \tag{2.23}$$

The discrimination loss enforces that the reconstruction $G(\mathbf{z}_\gamma)$ lies on the learned manifold of the normal data $\mathcal{X}$. It does so by feeding it to the learned discriminator network [113]:

$$L_D(\mathbf{z}_\gamma) = \sigma(D(G(\mathbf{z}_\gamma)), \alpha) \tag{2.24}$$

where σ is the sigmoid cross-entropy and $\alpha = 1$ the target label.

For the mapping to the latent space, the overall loss is defined as the weighted sum of the both aforementioned components [113]:

$$L(\mathbf{z}_\gamma) = (1 - \lambda) \cdot L_R(\mathbf{z}_\gamma) + \lambda \cdot L_D(\mathbf{z}_\gamma) \tag{2.25}$$

For the purpose of anomaly detection, anomaly score for the query sample $\mathbf{x}$ can be derived using the mapping loss function [113]:

$$A(\mathbf{x}) = (1 - \lambda) \cdot R(\mathbf{x}) + \lambda \cdot D(\mathbf{x}) \tag{2.26}$$

where the *residual score* $R(\mathbf{x})$ and the *discrimination score* $D(\mathbf{x})$ are defined by the residual loss $L_R(\mathbf{z}_\Gamma)$ and the discrimination loss $L_D(\mathbf{z}_\Gamma)$ at the last (Γ-th) update iteration of the mapping procedure. The model yields a large $A(\mathbf{x})$ for anomalous samples, whereas a small anomaly score means that a very similar sample to the queried sample was already seen during the training [113]. Once again, a threshold value needs to be introduced to classify input samples as anomalous or not.

While GANs can be used for anomaly detection in this fashion, the obvious drawback is that for every queried sample $\mathbf{x}$, Γ optimization steps have to be computed. This leads to an outstandingly bad test-time performance [116].

2.4.5 biGAN

The work of Donahue et al. [107] extends the GAN architecture with an encoder E, which maps input samples $\mathbf{x}$ to their latent representations $\mathbf{z}$. They named their model the **Bidirectional** GAN (BiGAN).

The encoder is a neural network like its counterparts the generator G and discriminator D. It's advantageous for the generator and the encoder to be each other's inverse, $E = G^{-1}$, and the authors claim that upon arriving at the global optimum, that is indeed the case. However, in practice the global optimum is rarely found and the generator and the encoder end up rather each other's approximate inverses [107]. Dumoulin et al. [117] explored various training strategies for the two networks to be each other's inverse mapping, and emphasized the importance of training the encoder jointly with the generator.

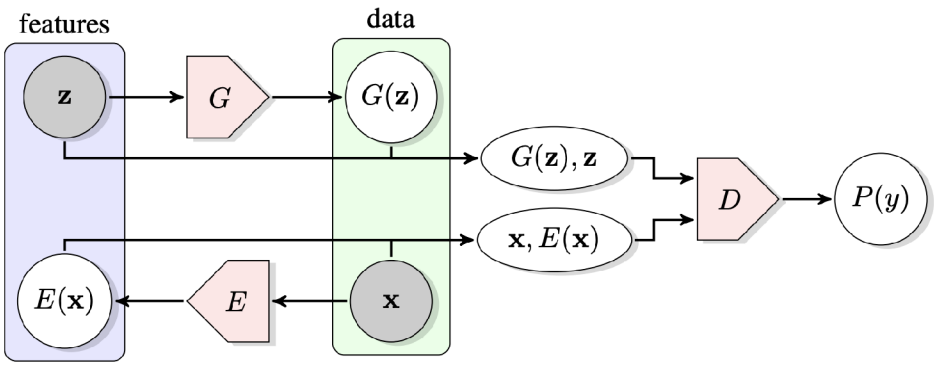


Figure 2.8: Illustration of the biGAN architecture. Courtesy of [107].

Another difference is that in this setting the discriminator also considers the latent representations, and hence must learn to classify pairs in the form of $(G(\mathbf{z}), \mathbf{z})$ or $(\mathbf{x}, E(\mathbf{x}))$. The biGAN training objective then becomes [116]:

$$\min_{G,E} \max_{D} V(D, G, E) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\mathbb{E}_{\mathbf{z}\sim p_E(\mathbf{z}|\mathbf{x})}[\log D(\mathbf{x}, \mathbf{z})]] \quad (2.27)$$
$$+ \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\mathbb{E}_{\mathbf{x}\sim p_G(\mathbf{x}|\mathbf{z})}[\log(1 - D(\mathbf{x}, \mathbf{z}))]]$$

Zenati et al. [118] successfully employed this architecture for the task of anomaly detection. Having trained the model in a semi-supervised manner, they used the same principle as the authors of anoGAN, even borrowing the anomaly score function:

$$A(\mathbf{x}) = (1 - \lambda) \cdot R(\mathbf{x}) + \lambda \cdot D(\mathbf{x})$$

where in the case of biGAN

$$R(\mathbf{x}) = |\mathbf{x} - G(E(\mathbf{x}))|$$

and

$$D(\mathbf{x}) = \sigma(D(\mathbf{x}, E(\mathbf{x})), 1)$$

The main advantage of using the biGAN model is that the anomaly detection process itself is now much faster, thanks to the trained encoder providing the necessary mapping, instead of having to rely on lengthy optimization [116, 118].

2.4.6 Recent applications

While generative models are a hot research topic of recent years and are achieving spectacular results in complex domains such as images or videos, their usage for detecting anomalies is not straightforward and the models themselves aren't trivial to grasp, and in the case of GANs not even to train.

In spite of that, they have found success detecting anomalies in medical [113, 119, 120, 121] or industrial [122, 123, 124] settings, as well as plant disease diagnosis [125], estimating physiological or psychological states by detecting anomalous facial skin temperature [126], security threat detection [127], detecting anomalous key performance indicators in web applications [128], network anomaly or intrusion detection [129, 130] or credit card fraud detection [131].

2.5 One-class neural networks

The most recent approach that leverages deep learning for anomaly detection is the **one-class Neural Network** (oc-NN). Proposed independently by Chalapathy et al. [132] and Ruff et al. [39], the oc-NN is

based the same idea as the hybrid models mentioned before. It shares the same advantages such as fast anomaly detection, extraction of informative complex features, being a classifier instead of outputting anomaly score or the capability of being trained in an unsupervised manner, through selecting a fraction of the training samples which can be considered as outliers by the model during the training [31, 39].

The key difference is that the oc-NN aims at mending the most notable shortcoming of the hybrid models, which is the failure to influence the representational learning of the neural network feature extractor with a suitable one-class classification objective. The oc-NN model on the other hand jointly trains the feature extractor network and the anomaly classifier using a custom one-class classification objective. The oc-NN is thus able to learn features useful for anomaly detection [31, 39].

Intuitively, instead of being composed of two independent parts like the hybrid approach, the oc-NN is a single neural network with a loss function resembling one-class classification objective of the traditional anomaly detectors. Chalapathy et al. [132] use a modified oc-SVM objective function for their design of the oc-NN while Ruff et al. [39] employ the SVDD objective. Both studies report state-of-the art results. The latter approach will be described further, to shed some more insight into the oc-NN designs.

2.5.1 Deep SVDD

The authors build on the kernel based SVDD by training a single neural network to jointly extract features useful for anomaly detection and to enclose the feature space representations of normal data with the smallest possible hypersphere [39]. They coin their architecture the **Deep SVDD**.

For input space $\mathcal{X} \in R^d$ and feature space $\mathcal{F} \in R^p$, let $\phi(\cdot; \mathcal{W}) : \mathcal{X} \rightarrow \mathcal{F}$ be a neural network with $L \in N$ hidden layers and set of weights $\mathcal{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_L\}$ where $\mathbf{W}_l$ are the weights of layer $l \in \{1, \ldots, L\}$. That is, $\phi(\mathbf{x}, \mathcal{W}) \in \mathcal{F}$ is the feature space representation of input sample $\mathbf{x} \in \mathcal{X}$ given by network ϕ with parameters $\mathcal{W}$. The aim of the Deep SVDD model is to learn the network parameters $\mathcal{W}$ together with minimizing the volume of the data-enclosing

hypersphere in feature space $\mathcal{F}$ that is characterized by radius $R > 0$ and center $\mathbf{c} \in \mathcal{F}$. Given training data $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the authors define the *souft-boundary* Deep SVDD objective as [39]:

$$\min_{R,\mathcal{W}} R^2 + \frac{1}{vN} \sum_{i=1}^{N} \max\{0, \|\phi(\mathbf{x}_i; \mathcal{W}) - \mathbf{c}\|^2 - R^2\} + \frac{\lambda}{2} \sum_{l=1}^{L} \|\mathbf{W}_l\|_F^2 \tag{2.28}$$

where as in traditional SVDD, minimizing R^2 minimizes the volume of the hypersphere, the second term is a penalty term for samples lying outside the hypersphere after being mapped to the feature space by the neural network, hyperparameter v serves the same role as explained in hybrid models, allowing a fraction of the training samples to fall outside the hypersphere, and the last term is a weight decay regularizer on the network parameters $\mathcal{W}$, with $\lambda > 0$ and $\|\cdot\|_F$ being the Frobenius norm [39].

Optimizing this objective allows the network to learn parameters $\mathcal{W}$ such that input samples are mapped closely to the hypersphere center $\mathbf{c}$. To achieve this, the network must learn to extract the common factors of variation of the data, which results in normal samples being mapped closer to the center $\mathbf{c}$ while anomalous samples are mapped further away. Minimizing the size of the hypersphere enforces this learning process [39].

The authors propose additional, simplified, objective to be used in cases, when the training data set consists of mostly or all normal samples, which is often the case in anomaly detection settings. The simplified objective is defined as [39]:

$$\min_{\mathcal{W}} \frac{1}{N} \sum_{i=1}^{N} \|\phi(\mathbf{x}_i; \mathcal{W}) - \mathbf{c}\|^2 + \frac{\lambda}{2} \sum_{l=1}^{L} \|\mathbf{W}_l\|_F^2 \tag{2.29}$$

This objective simply uses a quadratic loss for penalizing the distance of every given representation to $\mathbf{c}$. In this case, the volume of the hypersphere is minimized by mapping *every* sample close to the center $\mathbf{c}$, without penalizing the radius directly [39].

During anomaly detection, this objectives defines an anomaly score s for every input instance $\mathbf{x}$ as the distance of the point to the center $\mathbf{c}$ [39]:

$$s = \|\phi(\mathbf{x}; \mathcal{W}^*) - \mathbf{c}\|^2 \tag{2.30}$$

where $\mathcal{W}^*$ are the parameters of a trained network. For the soft-boundary version of the objective, the above equation can be enhanced by subtracting the final learned radius R^*. This leads to obtaining a classifier, which outputs a positive number for an anomaly outside the hypersphere and a negative number for a normal inlier [39].

Authors note, that the parameters $\mathcal{W}$ and the radius R often live on different scales and optimizing the network should be done in alternating steps, where one of the two is being optimized and the other remains fixed for several epochs and vice-versa. The authors also warn of possible *hypersphere collapse*, where the network learns the trivial solution of mapping every sample onto the center $\mathbf{c}$. To avoid this, they advise to follow these three principles [39]:

- Fixing the center $\mathbf{c}$ as the mean of the network representations, obtained by performing an initial forward pass on several of the training samples. This choice of $\mathbf{c}$ also leads to a faster and more robust convergence.
- Bias terms should not be used in Deep SVDD, as the model can learn the trivial solution using these biases.
- Unbounded activation functions (or functions only bounded by 0) should be preferred in Deep SVDD, such as the ReLU.

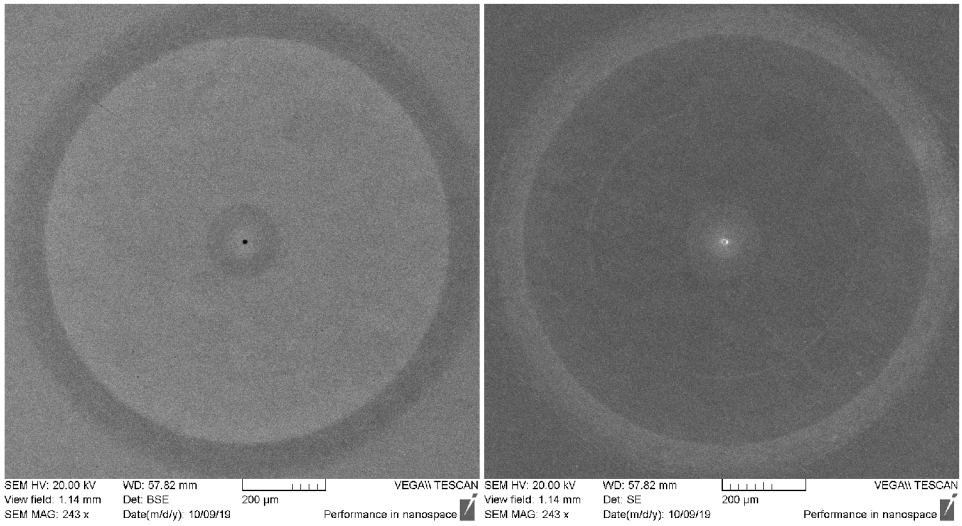
2.5.2 Recent applications

This approach showed great results on typical toy image data sets [39, 132], suggesting it might translate well into real world usage. Indeed, hybrid models have found practical success and oc-NN models build upon them and eliminate their one significant flaw. Yet because of the recency of this approach, only one practical application was found, in the domain of chemical process fault diagnosis [133].

3 Data set

The provided data set, courtesy of TESCAN, is made of images of electron-optical apertures, captured using a scanning electron microscope (SEM). An electron-optical aperture is a thin metal plate with a hole in its center, itself used in SEMs to control the size of the electron beam that passes through it. For it to work properly it has to be in perfect condition, without impurities and any surface damage.

The images come in two different variants for each aperture, based on the type of electrons being detected by the SEM, either the secondary electrons (SE) or the backscattered electrons (BSE). SE mode is useful for the inspection of the topography of the sample's surface, while BSE mode provides imaging that conveys information on the sample's composition. To delve deeper into the principles of electron microscopy is beyond the scope of this work.



(a) BSE
(Hi-res in B.1)

(b) SE
(Hi-res in B.2)

Figure 3.1: An illustration of a non-defective aperture in both modes.

Currently, a human expert has to go through each image and look for unacceptable impurities or damage. As hundreds of apertures are manufactured per year, this is a lengthy and costly process. Automated

anomaly detection would mean a significant reduction of costs and also would provide a shift from subjective decision making to objective one. Ideally, the employed model could also be retrained quickly, in the case of an emergence of new types of defects.

To research the possibility of automated anomaly detection, a data set of gray scale images was provided in two batches. The first batch consists of 275 (138 SE and 137 BSE) images of non-defective samples (normal data) and 46 (23 of SE and BSE each) images of defective samples (anomalous data). The normal samples are used for training purposes and the anomalous samples serve for testing mainly, but a portion of them is used when training deep distance metric learning models. The second batch consists of 69 (34 SE and 35 BSE) images of normal samples and none defective samples, and is used only for testing. With the exception of 2 apertures, each aperture has its SE and BSE mode image.

To make the images as uniform as possible, they have been rescaled to the minimal dimensions found among the provided data and the info bar at the bottom has been cropped, thus each image ended up with 768×768 resolution. Further details such as normalization are discussed in Chapter 4, as they differ within each approach.

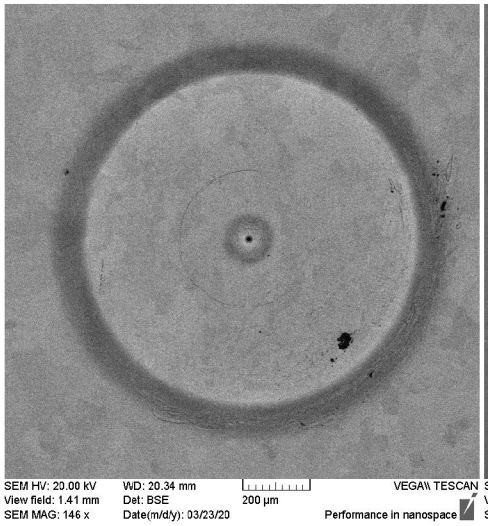
As the images are quite sizeable, Appendix B contains higher-resolution images of interest which would be difficult to place in the main body of the work.

3.1 Types of defects

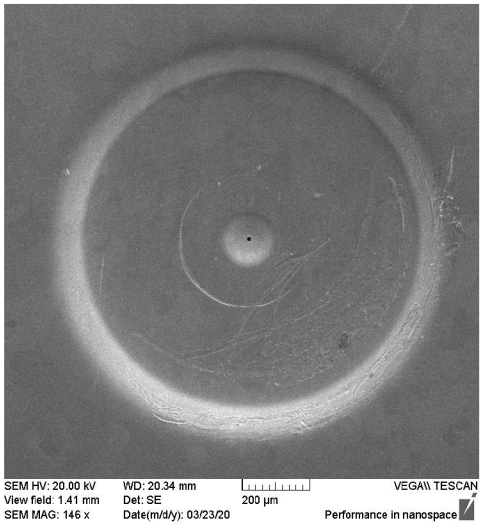
The provided anomalous samples introduce 3 types of defects. These are: **surface damage**, **stains** and **central hole filled with debris**.

Surface damage

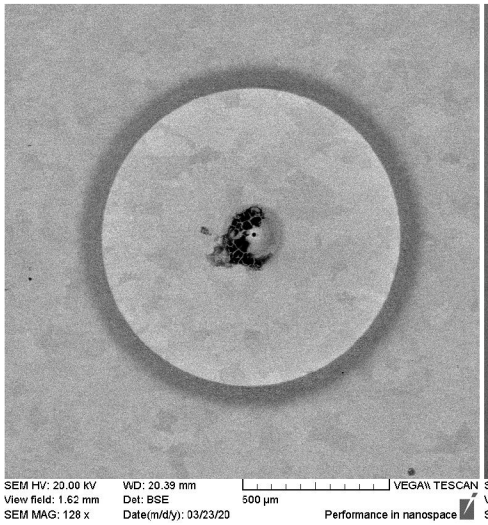
The most straightforward defect to spot and quite self-describing. Can be seen well in BSE images and possibly even better in SE images of original size. Showcase of two defective apertures follows.



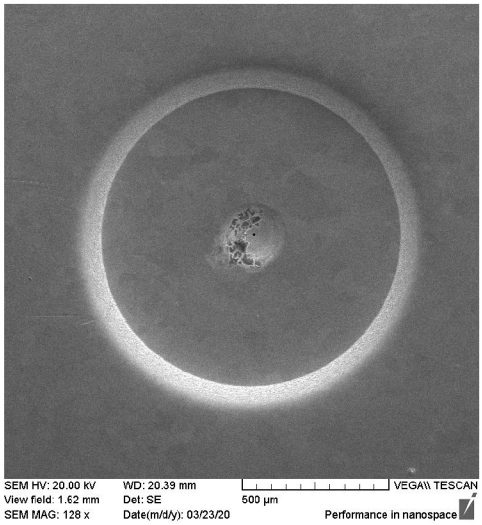
(a) BSE 1
(Hi-res in B.3)



(b) SE 1
(Hi-res in B.4)



(c) BSE 2
(Hi-res in B.5)



(d) SE 2
(Hi-res in B.6)

Figure 3.2: Two defective apertures with surface damage, in both image modes.

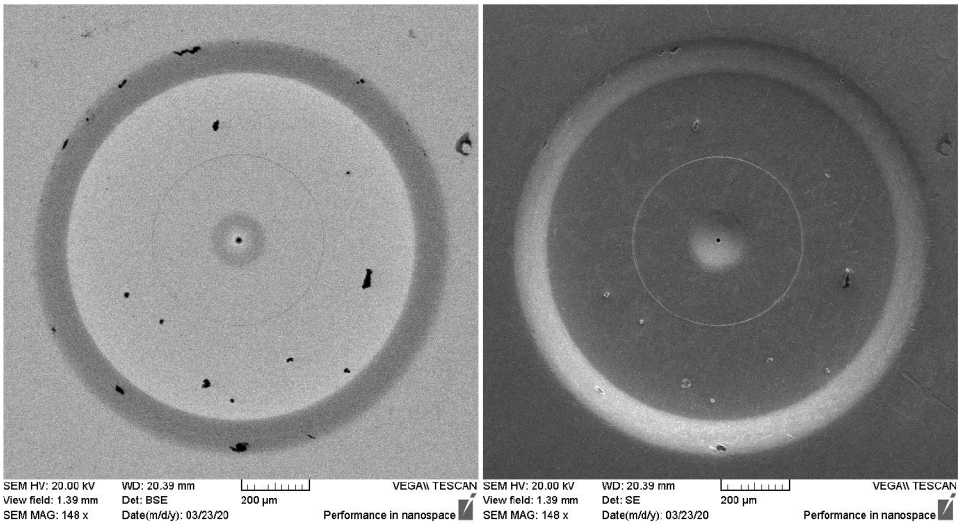
High-resolution images, providing better detail, can be found in Appendix B.

Stains

Stains are a more challenging type of defect due to the fact that some minor stains can be tolerated. However, there is no given rule describing what is tolerable and what isn't, and the classification is made by the judgement of a human expert. This might introduce inconsistencies over time.

Stains can be seen nicely on BSE images, manifesting as darker regions, while being slightly harder to spot on SE images, where they appear as gray or dark regions with a white outline.

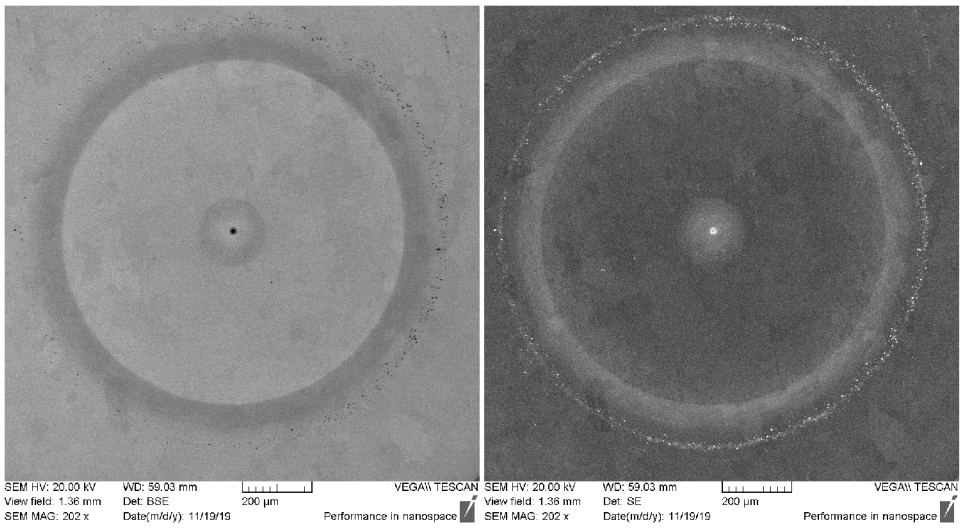
Showcase of a heavily stained, defective aperture ensues, followed by 2 non-defective apertures with tolerable stains, which often appear on normal samples. Finally, 4 apertures in BSE mode are shown to illustrate the thin threshold between normal and defective samples and the possible problem with subjective evaluation. As usual, high-resolution images appear in Appendix B.



(a) BSE
(Hi-res in B.7)

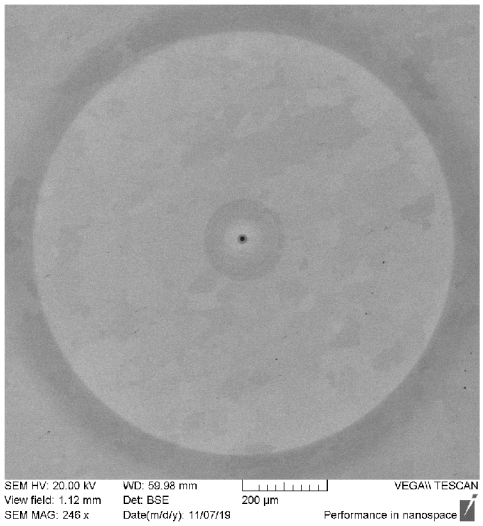
(b) SE
(Hi-res in B.8)

Figure 3.3: An illustration of a heavily stained, defective aperture in both modes.

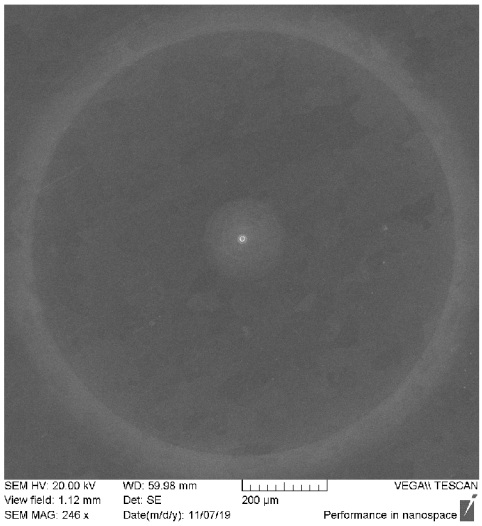


(a) BSE 1
(Hi-res in B.9)

(b) SE 1
(Hi-res in B.10)

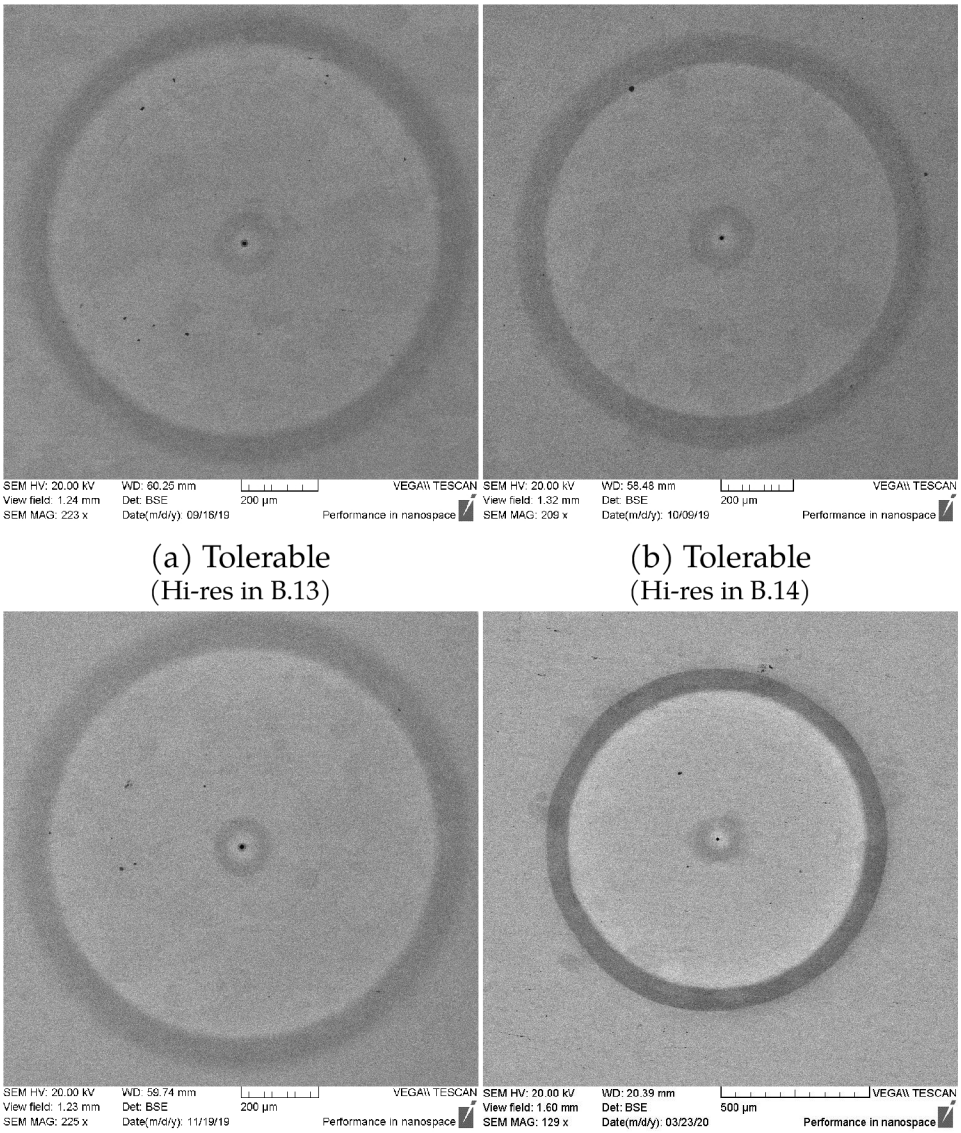


(c) BSE 2
(Hi-res in B.11)



(d) SE 2
(Hi-res in B.12)

Figure 3.4: An illustration of tolerably-stained apertures in both modes.



(a) Tolerable
(Hi-res in B.13)

(b) Tolerable
(Hi-res in B.14)

(c) Tolerable
(Hi-res in B.15)

(d) Defective
(Hi-res in B.16)

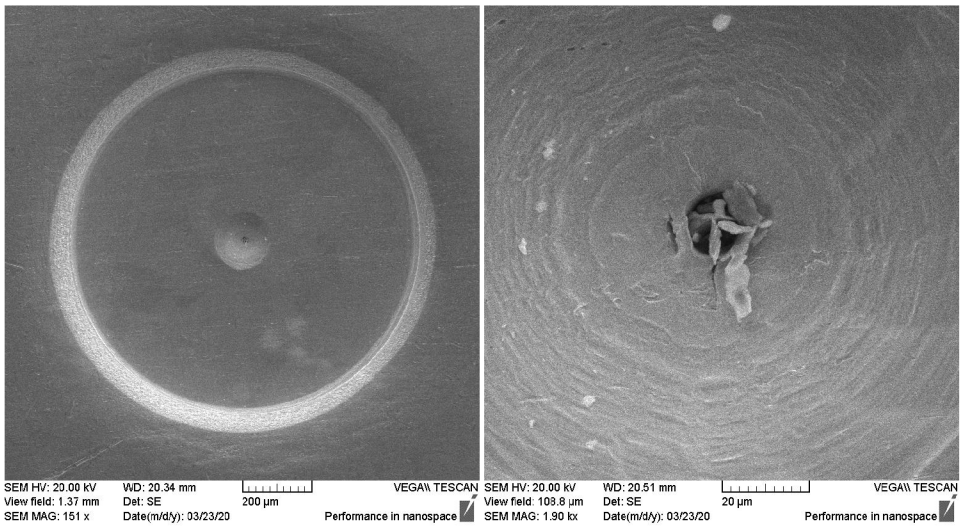
Figure 3.5: An illustration of the thin line between tolerable and defective stains.

Central hole filled with debris

The final and possibly the most difficult defect to detect. The central hole of the aperture can be clogged, but due to the small size of the hole even on a large image, it is extremely difficult to spot and a more magnified image of the hole must be procured.

Such magnified images are not a part of the given data set, with the exception of three apertures suffering from this type of defect. Two of the three apertures are also heavily stained, so the testing of this type of defect can hardly lead to anything conclusive.

While the defect *can* be spotted on SE images, BSE images hardly register it at all. This is illustrated further near the end of the Appendix B.



(a) Original SE
(Hi-res in B.17)

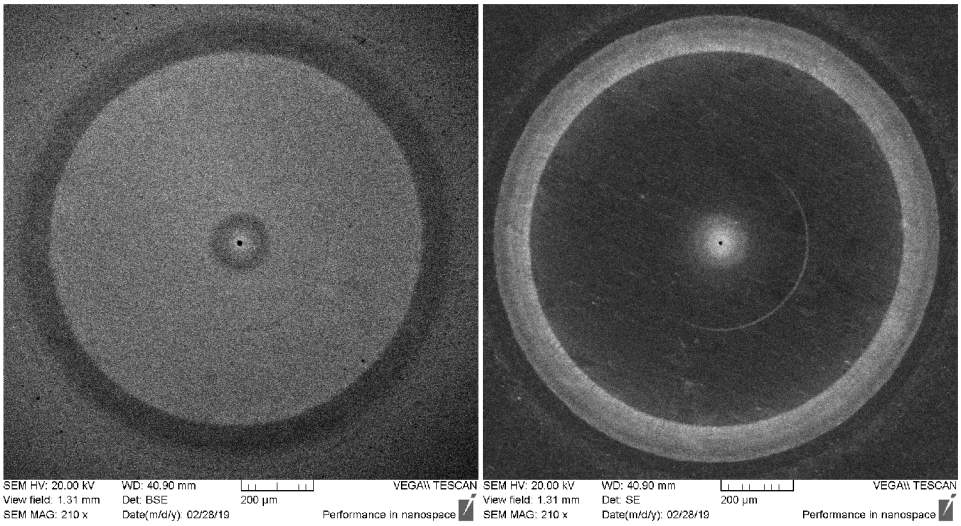
(b) Magnified SE
(Hi-res in B.19)

Figure 3.6: An illustration of a clogged aperture in SE mode.

3.2 Challenges

The main challenge of this data set stems from the relatively low number of samples, both for training purposes and for testing. Another complication is the variance of SEM parameters such as working distance, view field and magnification. The difference between images caused by different view fields and magnifications can be seen in the Figure 3.5, in the previous section describing stains.

Some combinations of magnification and working distance on the other hand produce noisy, low quality images, that also differ from most of the data set. A small number of non-defective images had to be removed entirely, because they were completely unusable, before arriving at the aforementioned number of 275 samples in the first batch.



(a) BSE (b) SE

Figure 3.7: An illustration of low quality images, differing from the majority.

Another challenge is the unclear threshold between acceptable minor stains and intolerable minor stains. This is further made difficult due to the differing magnification levels, therefore any trained anomaly detection model has to be capable of taking into account the relative size of the aperture when evaluating the size of the stains.

Finally, the images have to remain in high resolution when being fed to the model, as reduction comes with the risk of losing critical information, because stains are already quite small and there is a thin line between what is acceptable and what is not. Moreover, the clogging of the central hole translates into only a handful of pixels even on the 768×768 SE mode images. Most neural network models used for anomaly detection described in Chapter 2 are used on much smaller images, and therefore their performance on significantly larger image data is something of an uncharted territory.

4 Experiments

This chapter is dedicated to describing the experimental setting of each selected neural network-based approach chosen for the task of detecting defective apertures. As such, it explains approach by approach how the provided data set was used, the base architecture that was employed and how the anomaly detection process was carried out to obtain performance evaluation. The discussion of acquired results and how architecture modifications and hyperparameters affected the performance is presented in the following Chapter 5.

A total of 6 approaches were selected to try and find a model capable of detecting defective apertures. As the data set consists of image data, each described approach implicitly assumes that the neural network employed is of a convolutional variant. These 6 approaches are:

1. **Autoencoder**. The oldest and the simplest of the chosen models, it should serve as a baseline for comparisons. The main question is how well this model will handle discerning between acceptable and problematic stains, as it will probably learn to reconstruct the stains too well, since they appear even in normal data.

2. **Siamese network**. Deep distance metric learning (DDML) based anomaly detection showed promise in situations where the data set is rather small and imbalanced. This work is one such case, and simple Siamese network could prove usable.

3. **Triplet network**. An extension of Siamese network, the Triplet network is even more enforced to learn features that are key to similarity between normal samples, and could provide better results and faster convergence than the Siamese network in case that DDML is found useful.

4. **Hybrid approach with oc-SVM**. Hybrid model provides fast classification using the oc-SVM, which can be considered a significant advantage over interpreting custom anomaly scores and finding out appropriate thresholds. Possible question mark is the choice of the neural network to function as the feature extractor.

5. **Variational autoencoder**. VAEs have proven themselves to be capable anomaly detectors and it will be interesting to see if they can function well on this data set despite their reliance on reconstruction error as anomaly measure. And while the VAE architecture isn't the simplest to implement, there aren't many hyperparameters to tune compared to other generative models.

6. **BiGAN**. As GANs are quickly gaining traction in the field of anomaly detection, especially on image data, it would be appropriate to test them a bit outside of their usual setting of low-dimensional images, on a more challenging data set. The BiGAN architecture offers fast anomaly detection and the survey of Di Mattia et al. [116] has it outperforming other GAN architectures. The downside of GANs and a possible worry is their notoriously difficult training and lengthy hyperparameter tuning.

All of the neural network models are implemented from scratch using Python library Keras, while the oc-SVM used is a part of the scikit-learn framework.

The experiments themselves were run on the infrastructure provided by MetaCentrum NGI. Mainly on the *adan* cluster, where each node consists of 32× Intel(R) Xeon(R) Gold 5218 2.30 GHz CPUs and 2× nVidia Tesla T4 16 GB GPUs. One of each was used per single experiment. It should be noted that lesser GPUs struggle with batch size when using as high-dimensional data as in the case of this work. The prediction times stated later on for each approach are obtained querying a trained model using Intel(R) Core(TM) i5-9300H 2.40 GHz CPU.

The performance of each approach is rated by several criteria. The most crucial part of a model's performance is its ability to detect anomalies, which is rated in this work by the number of false positives (FP, normal samples classified as anomalous) and the number of false negatives (FN, anomalous samples marked as non-defective). Usually, the anomaly score decision threshold controls the trade-off between FP and FN. Ideally, there should be none of either case, but if such trade-off has to be made, the model should be tuned to minimize the number of FN while maintaining a reasonable number of FP, as defective apertures being passed as non-defective would be much

worse than the other way around. Using absolute numbers makes understanding the performance a bit more intuitive.

Another important aspect of each model's performance is the time it takes to classify a query sample as anomalous or not. Training time is of lesser concern as long as the model doesn't need to be retrained very often, which shouldn't be the case with any of the tested approaches. Hardware demands shouldn't be troublesome as well when it comes to prediction itself, as the memory footprint of each approach is fairly low (hundreds of megabytes maximum) and the computation done by a trained model isn't particularly taxing.

The difficulty of training of each model should be given some attention when it comes to overall evaluation, as well as the pros and cons of the actual anomaly detection process, which varies noticeably between the approaches.

Lastly, as both BSE and SE mode images are procured for each aperture, it's acceptable for the anomaly detection model to perform well only on either of the image types.

4.1 Autoencoder approach

4.1.1 Pure AE

The model of choice used in this approach is the convolutional autoencoder. The encoder part is a typical convolutional neural network, but when it comes to the decoder, there are two possibilities how to reconstruct the original sample from the features extracted by the encoder.

The simpler way is to use **UpSampling2D** layer, which essentially increases the size of the layer's input via interpolation and is not trainable. The other option is to use **Conv2DTranspose** layer, which performs transposed convolution and works in a similar manner to a convolution layer, also using filters, kernels and strides. The advantage of the second option is that the layer is trainable, but in practice its use is limited due to often producing a visible checkerboard artifacts on the reconstructed sample.

Both options are shown in the following tables, describing the architecture of the base autoencoder model used and improved upon.

Table 4.1: The composition of a single encoder layer, repeated several times in the encoder neural network.

Keras layer	**Size**	**Strides**	**Padding**
Conv2D	3×3	1×1	same
PReLU			
MaxPooling2D	2×2	2×2	same

Table 4.2: The composition of a single decoder layer, repeated several times in the decoder neural network, using UpSampling2D.

Keras layer	**Size**	**Strides**	**Padding**
Conv2D	3×3	1×1	same
PReLU			
UpSampling2D	2×2		

Table 4.3: The composition of a single decoder layer, repeated several times in the decoder neural network, using Conv2DTranspose.

Keras layer	**Size**	**Strides**	**Padding**
Conv2DTranspose	3×3	2×2	same
PReLU			

The PReLU layers are initialized with `alpha_initializer` $= 0.25$, the UpSampling2D layers use the *nearest* interpolation option. Notably, the base architecture does not use the Dropout or BatchNormalization layers.

The weights of all trainable layers are initialized using the Xavier initialization. The whole autoencoder is trained using pixel-wise MSE between the original sample and its reconstruction as the loss function, and the Adam optimizer with `learning_rate` $= 10^{-3}$ and otherwise default values.

The size of a single training batch is 4, as higher number leads to out of memory errors, and the number of epochs is not uniform,

but rather serves as a hyperparameter, further elaborated upon in the results chapter. The dimensions of the latent space representation are also examined as a hyperparameter.

Table 4.4: The architecture of the whole AE.

Layer	**Filters**	**Output shape**	**Alt. output shape**
Input		768×768	384×384
Encoder layer	32	384×384	192×192
Encoder layer	32	192×192	96×96
Encoder layer	32	96×96	48×48
Decoder layer	32	192×192	96×96
Decoder layer	32	384×384	192×192
Decoder layer	32	768×768	384×384
Conv2D (3×3, sigmoid)	1	768×768	384×384

The autoencoder is trained in a semi-supervised fashion, using only non-defective samples from the first provided batch with pixel values normalized into range $[0, 1]$. Several variants of the training data set are tested, namely using all of the 275 non-defective samples with the original resolution of 768×768, using only the 138 SE or the 137 BSE mode images (effectively specializing the model on one of the modes) with the original resolution, or using only the SE or BSE mode images with resolution reduced to 384×384, to see the effects of possible loss of detail.

To evaluate the usability of this approach as an anomaly detector, the trained model is tested on every available sample including the training set (as training does not guarantee there won't be any errors on the training set), and the testing set of sorts consisting of the defective samples and the additional non-defective images obtained in the second batch. Of course, only applicable images are tested, meaning that a model trained on BSE images only will be tested on BSE images only, too. Recall, that it is acceptable for the model to work only on one of the modes, as all apertures are scanned using both modes.

The anomaly detection itself is done in a typical fashion for the autoencoders, that is using the reconstruction error, which also serves as the loss function. This requires a decision threshold to divide the obtained reconstruction errors into acceptable and anomalous.

For this purpose, a threshold value of three times the standard deviation of the reconstruction errors obtained on the training data set is established. This comes from the empirical rule (also called the three-sigma rule) of statistics, which claims that for a normal distribution, almost all of the data will fall within three standard deviations. This rule is often used in outlier detection situations, and is a good initial threshold when no anomalous data is available to estimate a better one. As there are some anomalous samples available, the threshold value and its trade-offs are discussed further in the results chapter.

4.1.2 Ensemble with granulometry

As the pure AE is expected to struggle with classifying defects caused by stains due to its underlying principles, a simple ensemble extension is proposed.

Stains manifest on BSE images as noticeably darker regions than the rest of the aperture, therefore it could be possible to isolate them and distinguish between defective and acceptable measure of stains, using methods from traditional computer vision.

One such method is the **granulometry**, which is defined as the cardinality of an image obtained by performing morphological opening on a binary image using a structuring element. Detailed mathematical description is beyond the scope of this work, but intuitively, morphological opening removes regions of pixels that do not fully fit into the selected structuring element, and cardinality is then the sum of the remaining pixels.

In our case, the final sum could be used as an anomaly score. Certain small stains are tolerable, and using an appropriately sized (slightly larger than the allowable stains) structuring element, these stains would be erased and therefore would not contribute to the anomaly score, while large unacceptable stains would remain and would reflect upon the anomaly score. All that would remain is to find a suitable decision threshold.

Using this technique in tandem with the autoencoder model, one would obtain an ensemble of sorts, where each aperture could be rated by both techniques. In the case that their verdicts would not match, the aperture could be flagged for further review by a human expert. If perfect automatic anomaly detection could not be achieved, this could

at least lead to a noticeable reduction of the human expert's workload, provided that the ensemble learns to reliably classify the majority of the samples correctly.

To perform the granulometry and obtain an anomaly score, the queried image has to be thresholded into a binary image (inverted, so the dark regions become white and can be summed up). This is possible on BSE images, as stains are darker than anything else visible on the image, with the exception of the central hole, which has to be cropped off the image. Value serving as the binary threshold, size of the structuring element and the final decision threshold, to distinguish between normal and anomalous samples based on the resulting sum, all have to be found through experiments.

For implementation purposes the Scipy framework was used, providing the morphological opening operation, along with the OpenCV library, which provided easy access to different structuring elements and thresholding operations.

Using granulometry was tested on each available BSE mode image, both in original 768×768 and down-scaled 384×384 resolution, in tandem with the best performing AE model. For hyperparameter tuning, only the 137 non-defective BSE images were used. The ensemble's performance along with the best found hyperparameters are presented in the results chapter.

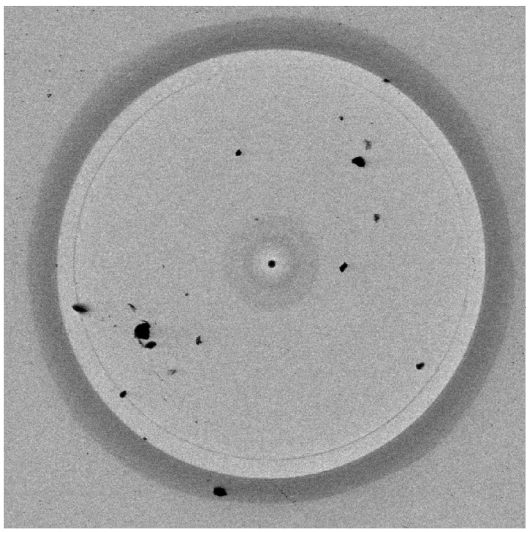


Figure 4.1: An original BSE image of a defective aperture.

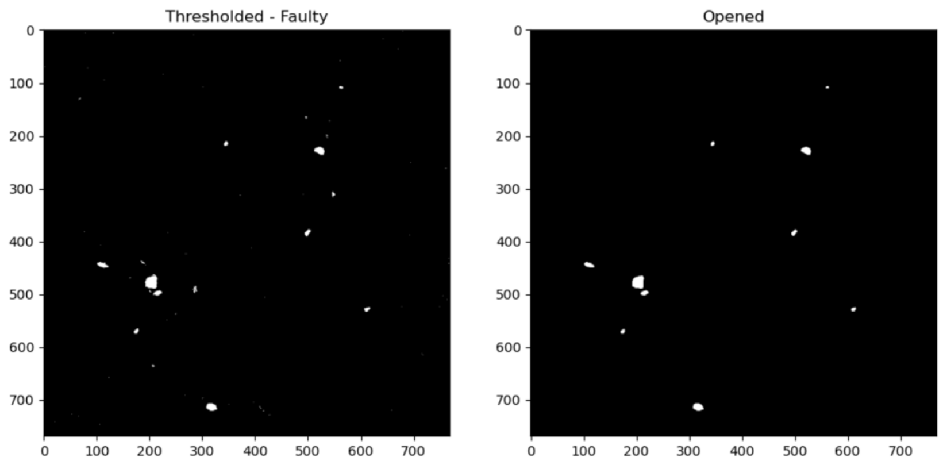


Figure 4.2: The same image thresholded (left) and opened (right). Large problematic stains remain and add to the anomaly score, while some lesser stains are erased.

4.2 Siamese network approach

To train and make use of a Siamese network, training pairs have to be created from the given data set. For training purposes, 80 % of the non-defective images from the first batch and 75 % of all the defective images are selected and normalized into range [0,1]. The pairs are created only from the images of the same type (BSE or SE), by pairing all of the selected non-defective images with each other and with all of the selected defective images. Anomalous-anomalous pairs are not made. This results in 7865 training pairs of SE mode images and 7739 BSE training pairs. Additionally, a label of 1 is assigned to normal-normal pairs and a label of 0 to normal-anomalous pairs. Defective images were chosen in such manner, so that at least 1 of each defect type is left for testing purposes. For each image type, both original 768×768 and reduced 384×384 resolution variants of training pairs are made.

The resulting training data set is indeed imbalanced, as many more pairs are of the normal-normal sort (about 76 % of the pairs). However, as stated when introducing DDML models, they have proven themselves capable of handling small and imbalanced data sets.

As with the AE approach, a batch size of 4 is used to prevent *out of memory* errors, and the number of training epochs is a hyperparameter, whose tuning is discussed in the according results section.

The base Siamese network architecture consists of two identical branches with shared weights, each being a convolutional neural network in this case. Detailed description of layers and their settings, which make up a single branch, is provided in the following tables. The Euclidean distance is computed between the outputs of the two branches using a custom Lambda layer, and finally a Dense layer with a single neuron and sigmoid activation provides a binary classification output.

The weights of all trainable layers are initialized using the Xavier initialization. Dropout rate is set to 0.5 and the BatchNormalization and ReLU layers are used with their default parameters.

To train the whole Siamese network, both binary cross-entropy and contrastive loss with margin of 1 are tested as the loss functions. Adam optimizer is used with `learning_rate` $= 10^{-5}$ and otherwise default settings.

Table 4.5: The composition of a single convolutional layer, repeated several times in one branch of the model.

Keras layer	**Size**	**Strides**	**Padding**
Conv2D	3×3	2×2	same
Dropout			
BatchNormalization			
ReLU			
MaxPooling2D	2×2	2×2	same

Table 4.6: Another type of composition of a single convolutional layer, with stacked Conv2D operations, as is commonly done in classification networks.

Keras layer	**Size**	**Strides**	**Padding**
Conv2D	3×3	2×2	same
Conv2D	3×3	2×2	same
Conv2D	3×3	2×2	same
Dropout			
BatchNormalization			
ReLU			
MaxPooling2D	2×2	2×2	same

Table 4.7: The architecture of the whole branch.

Layer	**Filters**	**Output shape**	**Alt. output shape**
Input		768×768	384×384
Convolutional layer	64	384×384	192×192
Convolutional layer	64	192×192	96×96
Convolutional layer	128	96×96	48×48
Convolutional layer	128	48×48	24×24
Flatten		294912	73728
Dense		64	64

Siamese networks themselves don't offer straightforward anomaly detection, therefore a method using *prototypes* was devised. The notion of using prototypes is inspired by the work of Freitag et al. [25]. The gist of this technique is to pair the queried sample with one or more hallmark samples and use them as the input to a trained Siamese network. Anomaly detection is then done based on the network's output per each comparison.

For this approach, 5 prototype images (for each image mode) of acceptable apertures have been hand-picked so that perfect apertures as well as slightly stained ones are represented. Custom anomaly score is proposed, where the queried image is tested against each of the prototypes, and when the model classifies them as similar, the score is increased by one, effectively stating how many prototypes the queried sample was found similar enough to. Therefore a score of 5 is desired for non-defective apertures and a score of 0 should be awarded to defective ones. As the network architecture contains a classification module with sigmoid activation, output > 0.5 is considered as *similar enough* result, consistently with the training labels, where 1 was assigned to normal-normal pairs. Scores between 0 and 5 are open for interpretation, and can for example be used to signal the need of manual decision by an expert. High resolution images of the selected prototypes can be found in Appendix C.

For performance evaluation, each available sample of applicable mode is tested against all 5 of the prototypes and anomaly score is obtained. Anything but score of 5 for non-defective samples or score of 0 for defective ones is considered as either false negative or false positive.

4.3 Triplet network approach

In a similar fashion to Siamese networks, a training set of triplets has to be created from the given data set. While online triplet mining was shown to improve the convergence speed, it is not applicable in this setting, as large batch sizes are required for it to work nicely. Instead, an alternative way of triplet construction is explored.

As with the Siamese network approach, 80 % of non-defective samples from the first batch and 75 % of available defective samples are selected and normalized into range [0,1]. Again, samples are divided by their image type, but this time only 384×384 resolution samples are used, as the Siamese network approach showed them to be the superior choice over the full-sized samples when training a DDML model.

Out of these selected images, one is randomly sampled to serve as the anchor of the triplet while a positive and a negative are also chosen at random (without allowing 2 identical images in 1 triplet) to complete the triplet. This process repeats until 5 000 training triplets are obtained. As this creation process involves the element of randomness, 3 training data sets for each image mode are created to observe its effect. Unlike pairs for Siamese networks, triplets do not require additional labels, as the label is effectively encoded into the triplet by its ordering. The obtained training data sets are more imbalanced than it is the case with the Siamese network approach, with usually about 700 triplets having a defective sample as the anchor.

The base model mostly shares the architecture of the base Siamese network, differing in being made of 3 identical branches instead of 2 and lacking the final layers that serve as a classification module. Instead, the output of the three branches is served straight into the Triplet loss function. The branches themselves are completely the same as in the Siamese network, hence their description lies in the Tables 4.5, 4.6 and 4.7.

The Xavier initialization is once again used for all the trainable weights, the dropout rate remains 0.5 and the BatchNormalization and ReLU layers are used with their default settings.

The Triplet network as a whole is trained using the Triplet loss, with margin to be enforced set to 10. Adam optimizer is used with

`learning_rate` = 10^{-4} and otherwise default Keras setting. The batch size remains at 4.

Detecting anomalies is done in the same vein as in the Siamese network approach, by using the same 5 prototype apertures and comparing every queried sample against them. As there is no classification module however, only a single branch of the trained model is used to extract features from the queried sample and a prototype, and squared Euclidean distance is computed between the feature vectors.

Using Triplet loss the model is trained to try to enforce a distance of at least 10 between the feature vectors of a non-defective and a defective sample. Since it is not given that the model will learn to do so perfectly, a decision threshold has to be used to classify the obtained distances.

For testing purposes, all of the available samples are used, each being tested against all of the prototypes. Anomaly score is given similarly as with the previous DDML approach. When the tested sample achieves a distance < 5.0 (the chosen threshold), it is considered as similar enough and 1 is added to the score. Total score of 5 again means that the tested sample was similar to all of the prototypes and can be considered non-defective, score of 0 implies a defective aperture. Any other score is considered as false negative or false positive.

4.4 Hybrid oc-SVM approach

To serve as a feature extractor, the encoder part of an autoencoder model or a single branch of a Siamese or Triplet network can be used. A league of such feature extractors was selected to be tested in a hybrid approach combined with the oc-SVM, in an effort to improve standalone performance (AEs) or to keep the excellent performance while improving evaluation speed (Siamese and Triplet networks). Best found models as well a diverse cast of slightly worse performers are among the selection, to see if perhaps the worse models can work better when coupled with a traditional classifier suited for anomaly detection. As the list is quite long, the selected models are described in Appendix D.

The feature extractors themselves are taken from already trained models from the three previous approaches. The oc-SVM of the hybrid

model is trained using the extracted features of the 275 non-defective samples from the first batch, divided by the image type. The radial basis function kernel is used, the parameter v, which controls the fraction of allowed false positives among the training samples, is tested across values $\{0.9, 0.7, 0.5, 0.4, 0.3, 0.25, 0.2, 0.1, 0.05, 0.02, 0.01, 0.001\}$, and the kernel coefficient γ is tested with both available options, *scale* and *auto*.

Anomaly detection is quite simple, the features of the queried sample are obtained using a neural network model and afterwards serve as the input to a trained oc-SVM, which outputs -1 for anomalies and 1 for non-defective samples. No thresholds are needed in this case.

For performance evaluation, all of the available samples, divided by the image type appropriately, are once again utilised. During training, the oc-SVM is allowed to falsely flag a fraction of the training non-defective samples as outliers, due to the parameter v. It does not necessarily have to, but since v cannot be set to 0, it can happen and is therefore an important performance gauge, as such false positives are unwanted.

The resolution of both the training and the testing data depends on the resolution on which the employed neural network of the hybrid model was trained on.

4.5 Variational autoencoder approach

In this approach, the neural network of choice is the most common kind of variational autoencoder, which uses a multivariate normal distribution as an approximation. It is the most popular VAE type as it allows for an easy sampling of the latent space, offers analytical evaluation of the KL-divergence term in the variational loss function, and the reconstruction term defaults to a MSE of the original sample and its reconstruction.

The model is trained in a semi-supervised manner, using only the 275 non-defective samples from the first batch, divided by their image type, normalized into range $[0,1]$. Only the 384×384 resolution of images is tested in this approach, as the previous approaches have confirmed this choice to be adequate. This resolution also allows to use a batch size of 16 during training.

The model's architecture resembles traditional autoencoder by being made of 2 parts, the encoder and the decoder. Again, two types of Keras layers (UpSampling2D and Conv2DTranspose) can be used in the decoder, and both are tested. However, the encoder of the chosen kind of VAE outputs two vectors representing the means and the variances of a multivariate normal distribution. The size of these latent representations becomes an important hyperparameter to tune. Additionally, a Sampling layer is required to start the decoding process. The following tables describe the architecture of the encoder and decoder parts, as well as the whole VAE.

Table 4.8: The composition of a single encoder layer, repeated several times in the encoder neural network.

Keras layer	**Size**	**Strides**	**Padding**
Conv2D	3×3	1×1	same
BatchNormalization			
PReLU			
MaxPooling2D	2×2	2×2	same

Table 4.9: The composition of a single decoder layer, repeated several times in the decoder neural network, using UpSampling2D.

Keras layer	**Size**	**Strides**	**Padding**
Conv2D	3×3	1×1	same
BatchNormalization			
PReLU			
UpSampling2D	2×2		

Table 4.10: The composition of a single decoder layer, repeated several times in the decoder neural network, using Conv2DTranspose.

Keras layer	**Size**	**Strides**	**Padding**
Conv2DTranspose BatchNormalization PReLU	3×3	2×2	same

Table 4.11: The architecture of the whole VAE.

Layer	**Filters**	**Output shape**
Input		384×384
Encoder layer	64	192×192
Encoder layer	64	96×96
Encoder layer	64	48×48
Encoder layer	64	24×24
Encoder layer	64	12×12
Encoder layer	64	6×6
Flatten		2304
Dense, Dense		12, 12
Sampling Lambda		12
Dense		2304
Reshape		$64 \times 6 \times 6$
Decoder layer	64	12×12
Decoder layer	64	24×24
Decoder layer	64	48×48
Decoder layer	64	96×96
Decoder layer	64	192×192
Decoder layer	64	384×384
Conv2D (3×3, sigmoid)	1	384×384

All of the trainable weights are initialized with the Xavier initialization. As with the classical AEs, PReLU layers are initialized with `alpha_initializer` = 0.25 and the UpSampling2D layers use the *nearest* interpolation option. BatchNormalization layers are used with their default settings.

The whole VAE is trained using the variational loss, with the reconstruction loss factor $a_{rl} = 10^3$. Adam optimizer is employed with `learning_rate` $= 5 \times 10^{-4}$ and otherwise default settings. The size of the latent vectors (the encoder outputs) begins at 12.

For the task of anomaly detection, reconstruction error given as the MSE between the queried input and its reconstructed output is used, much like with the classical AE approach. Therefore a decision threshold is needed to classify the queried sample as defective or not.

To gauge the performance of this approach, all of the available samples divided by their image type are tested. Decision threshold based on the empirical rule of statistics serves as an estimate, again exactly as in the classical AE approach.

4.6 BiGAN approach

The most challenging of the approaches, both architecture and training-wise. GANs as anomaly detectors have been reliably used on image data of fairly low dimensions ($< 128 \times 128$), therefore it's an interesting venture to try them on the provided high-dimensional data set.

Because of such limitations, images of 384×384 and, uniquely among other approaches, 192×192 resolution are used for training and testing. Training itself is done in a semi-supervised manner, where all of the 275 non-defective samples of the first batch, again divided by their image type, serve as the training data set. The lower resolution allows for a batch size of 16. Notably, the samples are normalized into range [-1,1] this time, as this reportedly improves the performance of GANs.

As is usual with GANs, the discriminator network and the encoder and generator networks are taking turns during training. By default, 1 discriminator training epoch is followed by 1 encoder + generator training epoch, although asymmetric training is also tested. The discriminator training epoch consists of training on a batch of generated (fake) samples followed by a batch of real samples. Using such exclusive batches was shown to improve the discriminator's training.

Both the Wasserstein and the binary cross-entropy losses were tested for each of the two alternating trainings. As the Adam opti-

mizer is not recommended when using the Wasserstein loss to train the discriminator [115], SGD is picked instead, with Adam used to train the generator and encoder. The learning rate for both of the training processes is set to 5×10^{-4}, and Adam's $\beta_1 = 0.5$. Other optimizer settings remain default. When using the Wasserstein loss, the Conv2D and Dense layers in the discriminator network have their weights clipped into range [-0.01,0.01] after each update. The default used latent space is 24-dimensional, with each variable drawn from a normal distribution.

The architectures of all of the three networks, which comprise the BiGAN model, are described in the following tables. This time, only UpSampling2D layer is used in the generator, as Conv2DTranspose proved to be inadequate in previous approaches that tested it, due to producing checkerboard patterns.

The BiGAN discriminator network is made of two branches. This is due to the input being an image + latent representation pair. Typically, the branch dealing with the image input is of a convolutional neural network form, while the branch for the latent representation uses Dense layers. The two branches are later flattened and concatenated, with the final Dense layers serving as the classification module.

Table 4.12: The composition of a single encoder layer, repeated several times in the encoder neural network.

Keras layer	**Size**	**Strides**	**Padding**
Conv2D	3×3	1×1	same
LeakyReLU			
MaxPooling2D	2×2	2×2	same

Table 4.13: The composition of a single generator layer, repeated several times in the generator neural network.

Keras layer	Size	Strides	Padding
Conv2D	3×3	1×1	same
LeakyReLU			
UpSampling2D	2×2		

Table 4.14: The composition of a single convolutional layer, repeated several times in the image branch of the discriminator network.

Keras layer	Size	Strides	Padding
Conv2D	3×3	2×2	same
BatchNormalization			
LeakyReLU			
Dropout			
MaxPooling2D	2×2	2×2	same

Table 4.15: The architecture of the whole encoder network.

Layer	Filters	Output shape	Alt. output shape
Input		384×384	192×192
Encoder layer	32	192×192	96×96
Encoder layer	32	96×96	48×48
Encoder layer	32	48×48	24×24
Encoder layer	32	24×24	12×12
Encoder layer	32	12×12	6×6
Encoder layer	32	6×6	3×3
Flatten		1152	288
Dense		256	256
LeakyReLU			
Dense		24	24

Table 4.16: The architecture of the whole generator network.

Layer	**Filters**	**Output shape**	**Alt. output shape**
Input		24	24
Dense		1152	288
Reshape		$32 \times 6 \times 6$	$32 \times 3 \times 3$
Generator layer	32	12×12	6×6
Generator layer	32	24×24	12×12
Generator layer	32	48×48	24×24
Generator layer	32	96×96	48×48
Generator layer	32	192×192	96×96
Generator layer	32	384×384	192×192
Conv2D (3×3, tanh)	1	384×384	192×192

Table 4.17: The architecture of the whole discriminator network.

Layer	**Filters**	**Output shape**	**Alt. output shape**
Latent			
Input		24	24
Dense		256	256
LeakyReLU			
Dense		256	256
LeakyReLU			
Image			
Input		384×384	192×192
Convolutional layer	64	192×192	96×96
Convolutional layer	64	96×96	48×48
Convolutional layer	128	48×48	24×24
Convolutional layer	128	24×24	12×12
Flatten		73728	18432
Joint			
Concatenate		73984	18688
Dense		256	256
LeakyReLU			
Dense		1	1

All of the trainable weights are initialized using the Xavier initialization. The slope of all LeakyReLU layers is set to 0.1, and the used dropout rate is 0.2. UpSampling2D layers again use the *nearest* interpolation option. All of the other possible layer settings are left default. The final Dense layer of the discriminator network uses sigmoid activation when the model is trained with binary cross-entropy, otherwise its output is raw.

For the task of anomaly detection, anomaly score consisting of reconstruction and discriminator loss is computed for the queried sample, exactly as proposed by the authors of anoGAN and EGBAD. The reconstruction loss is given as the MSE between the original sample and its reconstruction, obtained by first encoding the sample into the latent space and then reconstructing it using the trained generator network. The discriminator loss is simply the output of the discriminator/critic network when the queried sample and its latent representation, obtained by using the trained encoder, are fed as the input. By default both of the terms are weighted equally.

The trained models are tested on all of the available non-defective and defective samples, divided by their image type. As an anomaly score is used, a decision threshold for sample classification must be stated. When binary cross-entropy is used as the loss function, three times the standard deviation of the anomaly scores obtained on the training data set is a sufficient threshold estimate, which can then be edited. However, this threshold does not work nicely with the unbounded Wasserstein loss, therefore a suitable threshold must be found manually after the training concludes.

5 Results

This chapter presents the best achieved results per each approach. Findings about how architecture and hyperparameter choices influence the performance are also discussed, as is the training time and stability, as well as the types of defects which have proven to be problematic.

As a lot of modifications to the base architectures were tested in each approach, graphs of the anomaly detection performance and the training loss and accuracy (where applicable) were saved for each tested model and are a part of the electronic appendix. The weights and the whole trained models were also saved, but due to their size only the best ones made it to the electronic appendix. Saving only the weights provides lower overall size, but saving the whole trained architecture (which includes the weights too) removes the need of including the model classes in the anomaly detector.

5.1 Autoencoder approach

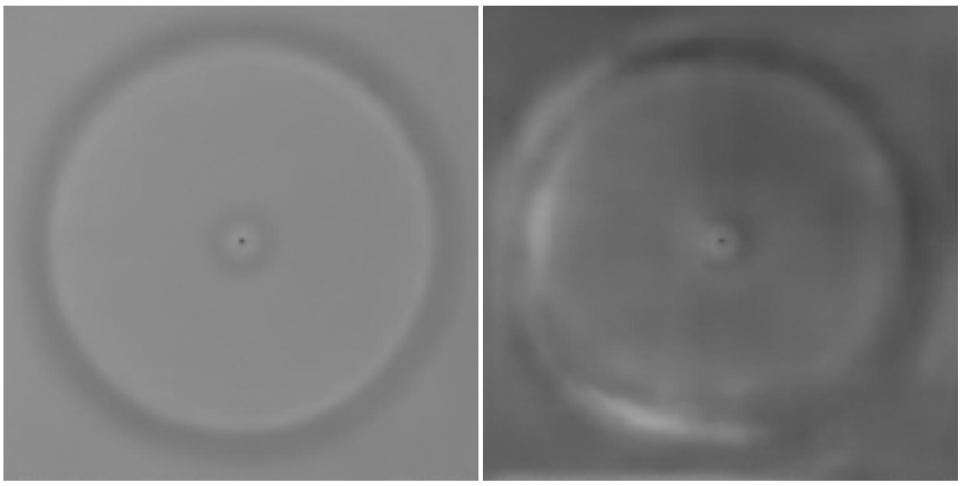
5.1.1 Pure AE

The base model showed hints of usability, but ultimately was not good enough for any form of practical detection. Therefore, several adjustments to hyperparameters and architecture had to be tested. Two critical factors were found, which together lead to the best performing model.

The first, slightly more important one, is the depth of the encoder and decoder networks. Too shallow networks led to nice reconstruction, but also to poor anomaly detection performance, as the autoencoder learns to reconstruct stains and even surface damage far too well. It is important to limit the latent space only to the most important features. This is done by making the encoder and decoder networks deeper, as each encoder layer, presented in the previous chapter, reduces the dimensions of the input sample, therefore stacking more of them leads to a smaller latent space. Decoder network then has to match this depth to reconstruct the input. The reconstruction suffers in quality, but there is not enough space left for the model to learn

to reconstruct unwanted features, which leads to noticeably larger reconstruction errors in defective samples. The second factor is the learning rate, which by itself is not good enough to improve the base architecture significantly, but is very important in getting the best out of the deeper models.

The best found model uses 8 encoder and decoder layers, with a learning rate of 10^{-5}. Reducing the learning rate further led to worse performance. The latent space is reduced to a mere $32 \times 3 \times 3$, and the reconstructions resemble a blurry, stainless aperture. Defective samples show apparent noise caused by the inability to reconstruct heavy staining or surface damage.



(a) Non-defective sample. (b) Sample with damaged surface.

Figure 5.1: An illustration of reconstructions done by the best found, deep, model.

This best performing model produces 21 false positives (out of 344 tested non-defective samples) and 6 false negatives (out of 46 defective samples). All of the false negatives are BSE type images, where the stains are rather small and border on acceptable. The false positives are mostly images which ended up darker, blurrier, or noisier when compared to the rest of the samples, due to different SEM settings during their capture (see Figure 3.7). Other portion of the false positives are non-defective images, which border on being defective due to stains

(see 3.5). Surprisingly, the threshold estimate proved an extremely nice choice, with further manipulation basically unnecessary.

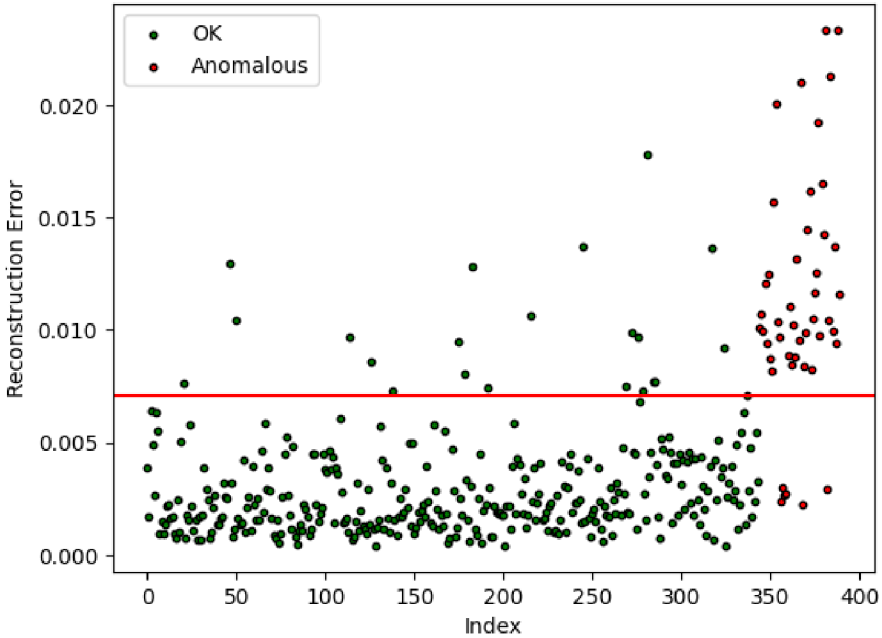


Figure 5.2: The performance of the best found model. The reconstruction error of each of the tested images is plotted, with the red line being the estimated decision threshold, described in the previous chapter. Reconstruction error above this threshold implies an anomaly. Note that 2 of the dots representing false negatives merged due to being next to each other and with similar score, hence it looks like there are only 5 false negatives.

Interestingly, training the model on only one image type produces basically the same results as training it on a combined training set. Using lower-dimensional data leads to an increase in false positives, as was expected, due to information loss. The training itself is rather stable, and training the model several times leads to almost identical results. The ideal number of epochs was found to be between 600 and 800, with more than that leading to no further improvement. The training time is then around 3 hours, which is very swift.

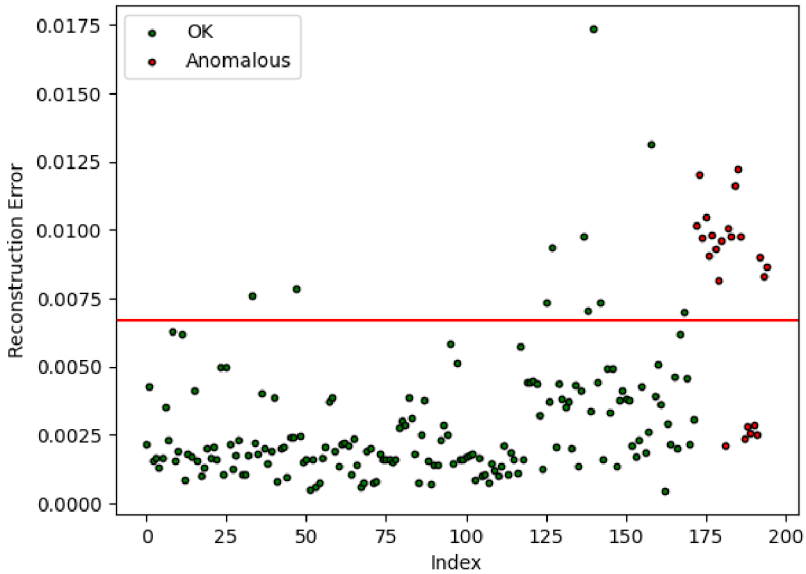


Figure 5.3: The same best performing model, trained on exclusively the BSE type images.

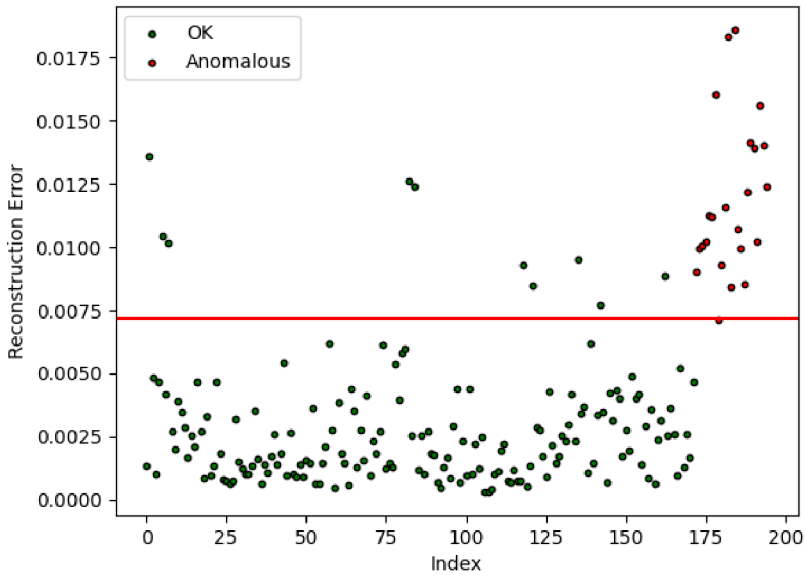


Figure 5.4: The same best performing model, trained on exclusively the SE type images.

As for other interesting hyperparameter and architecture tuning choices, adding any sort dropout leads to performance deterioration and should be avoided when using an autoencoder model. The number of filters in Conv2D layers matters too, as increase to 64 or decrease to 16 led to a few more false positives and negatives. Stacking several Conv2D layers also led to slightly poorer performance, as did increasing the kernel size. UpSampling2D layer in the decoder network is highly preferred, as using Conv2DTranspose in otherwise the same model always caused far worse results.

Overall, the results showed expected difficulties with the threshold between the acceptable and defective amount of stains. However, the autoencoder model can work very well on SE images, as it detected all of the defective samples and the amount of false positives could be reduced by uniform SEM settings during the image capture. At the very least, it could be used in a setting where any sample with reconstruction error above the customizable decision threshold will be flagged for further inspection. If most of the apertures turn out non-defective, this could lead to a significant reduction of effort on the side of the human expert.

Other benefits of this approach include a very fast, stable training, ease of implementation, and a reasonably fast detection time of around a third of a second per queried sample.

As for BSE images, there is still the possibility of human error during the labeling of images, so it might be worthwhile to revisit the false positives and negatives and reach a conclusion afterwards. Detector script which shows images classified as a false positive or negative is included in the electronic appendix.

5.1.2 Ensemble with granulometry

For this enhancement to work, several hyperparameters had to be tuned on the training set of BSE images. The optimal value for binary thresholding ended up being 55, the best structuring element is a circle with a diameter of 4 px and the decision threshold for sample classification produced the best results when set to 50. These values are for images with resolution of 768×768, the best found values for lower resolution images performed significantly worse, as no good

enough structuring element could be found. Sub-pixel structuring elements [134] could possibly lead to improvement in that matter.

Using these values and structuring element, granulometry helps with correctly detecting 3 of the false negatives out of the 6 that the best autoencoder model fails on. All the while introducing only 1 additional false positive, which is a rather dark image when compared to the rest of them, caused by the differing SEM settings. This false positive could therefore be amended by uniform SEM settings for each sample. Granulometry itself is very fast, bringing the overall detection time to just slightly over a third of a second per sample.

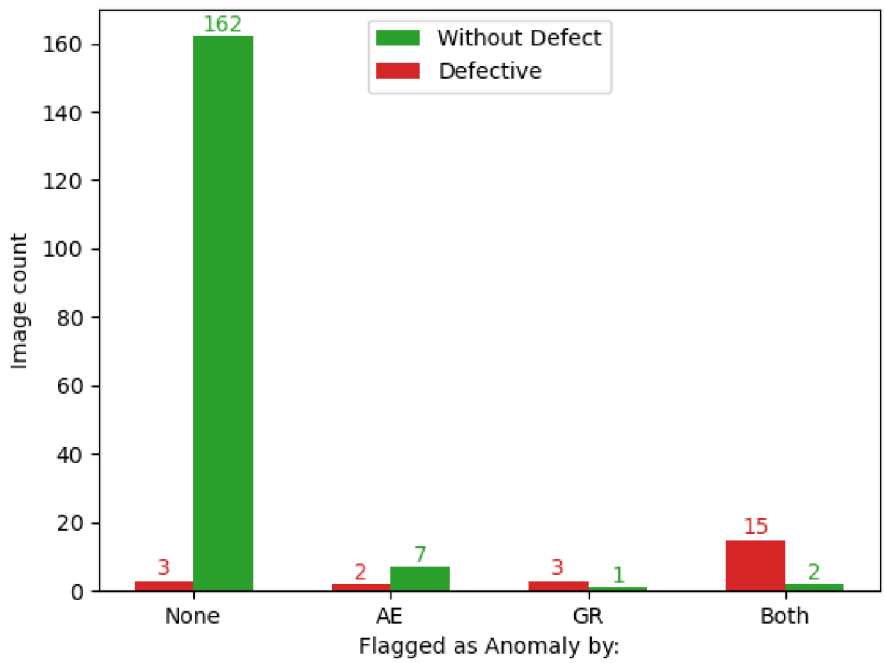


Figure 5.5: The result of using the best performing autoencoder model together with granulometry as a detection ensemble. Each column represents the number of samples flagged as anomalies by: 1) none of the techniques, 2) autoencoder, 3) granulometry, 4) both.

In fact, all 3 of the false positives the granulometry incorrectly classifies (1 unique to it, 2 common with the autoencoder) are way too dark images and the failure stems from the binary thresholding.

Adaptive thresholding was hence tested to try to amend this shortcoming, but no hyperparameter settings could be found that would outperform the binary variant.

The three remaining undetected false negatives are all images with very small stains, which could raise the question if perhaps they were misclassified by the human expert, as very similar images can be found among the non-defective sort in the data set. If indeed they were, this ensemble approach could reduce the workload of a human expert by only leaving images not flagged as anomalous by either of the techniques for further inspection.

While this enhancement improves the performance of the autoencoder model on BSE type images, it still does not match the results obtained using only the autoencoder on SE images, unless the three false negatives were misclassified when labeled.

5.2 Siamese network approach

The performance of Siamese models vastly differed based on the image type the model was trained and tested on, and the resolution of images. Using either the base or the stacked variant of convolutional layer in the branches produced mostly equal results when measured by false positives and negatives, with stacked convolutions showing slightly better prediction confidence most of the time. What proved to be key in the pursuit of the best performance for each combination of the models and image types however, was precise depth of the branches, learning rate and the removal of dropout.

When using the simple convolutional layer, the base depth of 4 layers in a branch proved to be superior, as deeper or shallower branches vastly deteriorated the performance. For the stacked variant, slightly shallower depth of 3 performed the best. The tested models were very sensitive when it comes to the chosen learning rate, and the base value of 10^{-5} produced by far the best results. Finally, the removal of dropout was crucial for getting the most out of the BSE-trained models. And while not needed, it improved the confidence of SE-trained ones.

When it comes to the image type used for training and testing, models trained on SE variant reached perfect anomaly detection (0 FP

and FN) quite easily, were resistant to architecture and hyperparameter changes (with the exception of branch depth and learning rate) and their training was very stable, producing the same results after each run. Image resolution also played no significant part, but lower-dimensional images did lead to a better prediction confidence, similarly to dropout removal. For both tested resolutions, the best models found employed the stacked variant of convolutional layers in its branches, but the simpler variant wasn't worse in anomaly detection performance, only slightly in the prediction confidence.

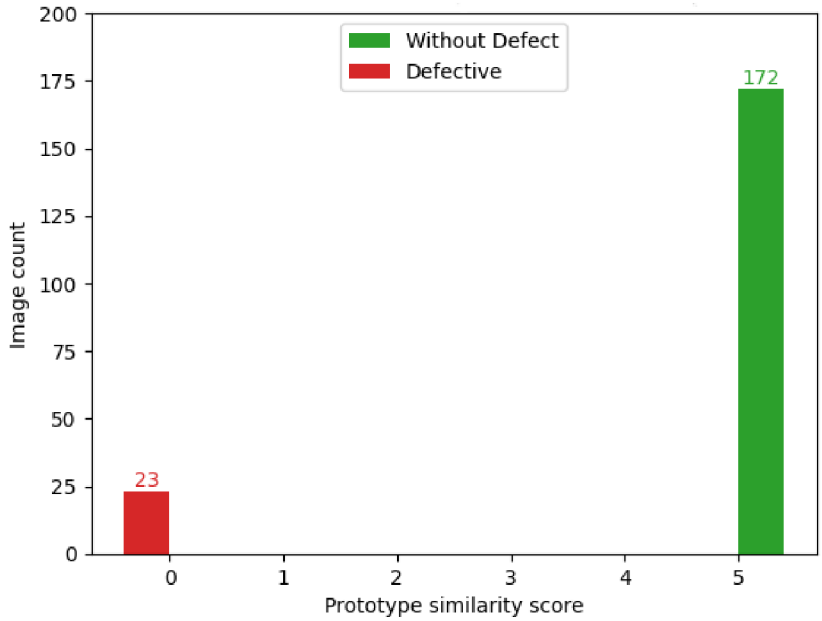


Figure 5.6: The performance of the best found SE-trained model, using the stacked variant of convolutional layer and absent dropout, trained for 30 epochs on 384 × 384 resolution images. Each column represents the number of samples assigned to a score. As described in the previous chapter, this score serves as a similarity measure of the sample to the 5 chosen prototypes. Hence, a score of 5 is desired for non-defective samples and score of 0 for defective ones.

As for the performance on BSE images, much tuning was required, as the models struggled to learn to detect anomalies perfectly. Removal of dropout proved essential, as did finding the optimal branch

depth. While the capabilities of SE-trained models were not affected by changes to the number of filters in Conv2D layers and neurons in Dense layers, this was not the case with BSE-trained ones, as any increase or decrease, compared to the base values, negatively influenced their performance. On 768 × 768 images, even the best models could not reach perfect detection, frequently scoring between 4 to 7 false negatives and 0 to 2 false positives. These intervals are given due to repeated training runs leading to slightly different results, therefore to obtain the best possible model, several training runs had to be done and the best one chosen afterwards. This training fluctuation is another stark difference to SE-trained models. Upon using the 384 × 384 images however, the best found BSE-trained model did indeed reach perfect anomaly detection of 0 FP and FN. Again, it used the stacked variant of convolutional layers.

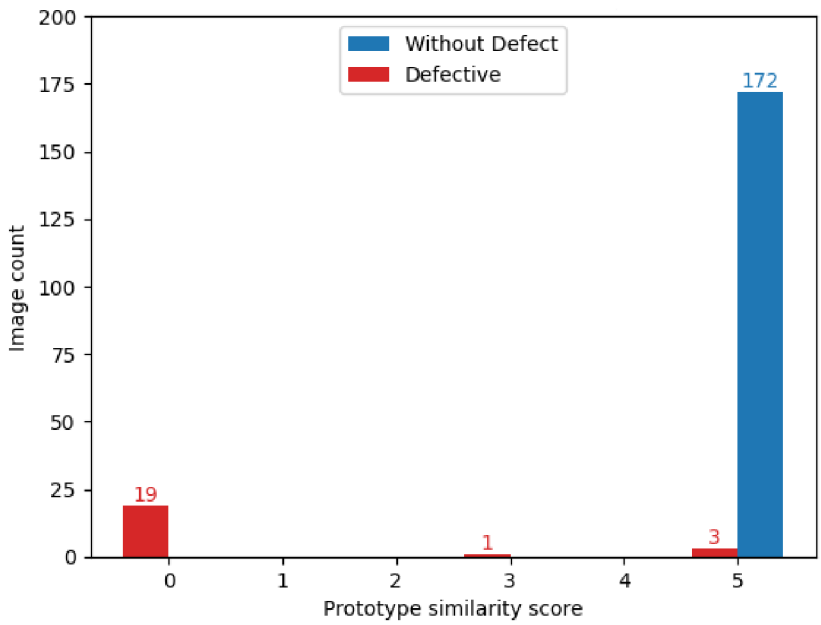


Figure 5.7: The performance of the best found BSE-trained model on high resolution images, using the base architecture without dropout, trained for 40 epochs.

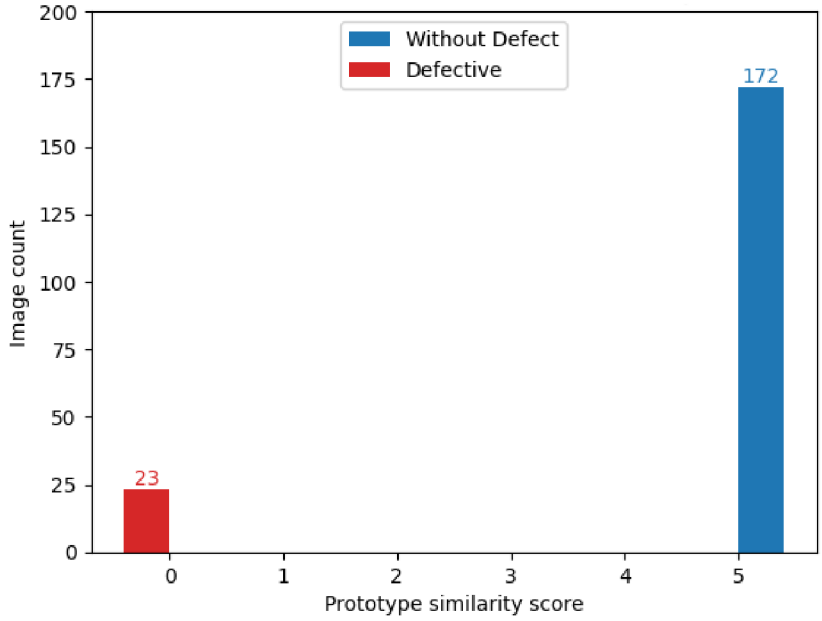


Figure 5.8: The performance of the best found BSE-trained model on lower resolution images for 30 epochs, using the stacked variant of convolutional layer and without dropout.

The best number of training epochs for models trained on either of the image types was shown to be between 30 to 60, with generally lower number of epochs being sufficient for SE-trained models and higher number needed for their BSE counterparts. Higher number of epochs than 60 usually led to overfitting.

Interestingly, using the lower resolution images (384×384) led to the best overall obtained performances and improved the training stability. Even the base architecture without additional improvements reached formidable results for both of the image types. Reducing the resolution even further led to a drop in performance though, as too much detail was lost.

Additional advantage of lower-dimensional images lies in the training and detection time. Training the Siamese models proved costly in memory and time resources, with 768×768 images requiring upwards of 15 hours of training time and 6 seconds of detection time per sample. Contrastingly, using 384×384 resolution leads to training time comparable with autoencoders, of about 5 hours, and detection

time of 1 second per queried sample. The optimal performance is therefore slightly offset by the anomaly detection time.

The choice of employing either binary cross-entropy or the contrastive loss proved of little consequence, as both losses produced fairly equal results in this scenario, with contrastive loss generally showing miniscule increase in prediction confidence of SE-trained models and slight increase of false positives when used with non-perfect BSE-trained models.

Unlike in the autoencoder approach, the models did not struggle with the same defective or non-defective samples. Most of the false negatives or positives were again samples with stains very close to the acceptable threshold, but the concrete samples varied between each training run.

This approach showcased the power of Siamese networks in anomaly detection scenarios where at least a few anomalies are available for training purposes. Despite the class imbalance present in the training data sets, the best performing models for each of the two image types did indeed learn to classify the available defective and non-defective samples with optimal accuracy. SE images were shown to be the easier of the two for the models to master, while obtaining the perfect performance on BSE images required additional tuning effort.

As images of each type are captured for every aperture, both the best BSE and SE-trained models could perhaps be used in tandem, to further strengthen their capabilities by reporting any aperture that obtains differing verdicts for manual inspection.

The obvious downside of this approach is lengthier detection time per sample. Long retraining is a lesser issue, as Siamese networks were shown to perform very well even when additional classes of anomalies are introduced, therefore retraining can be avoided.

5.3 Triplet network approach

Following the success of the Siamese network models on lower resolution images, Triplet networks were trained and tested exclusively on images of the same lower resolution. The base architecture by itself showed promise on SE images, scoring less than 10 false negatives and 0 false positives on the testing set, depending on the training variance. However, it proved unsuitable for BSE images, being unable to detect anomalies in any capacity.

The single critical factor of this approach proved to be the dropout, similarly to the Siamese networks. While only lowering it improved the performance of the base architecture on SE images to only 3 false negatives (with no effect on BSE-trained models), the complete removal of dropout layers led to achieving perfect anomaly detection of 0 false positives and negatives for both image types.

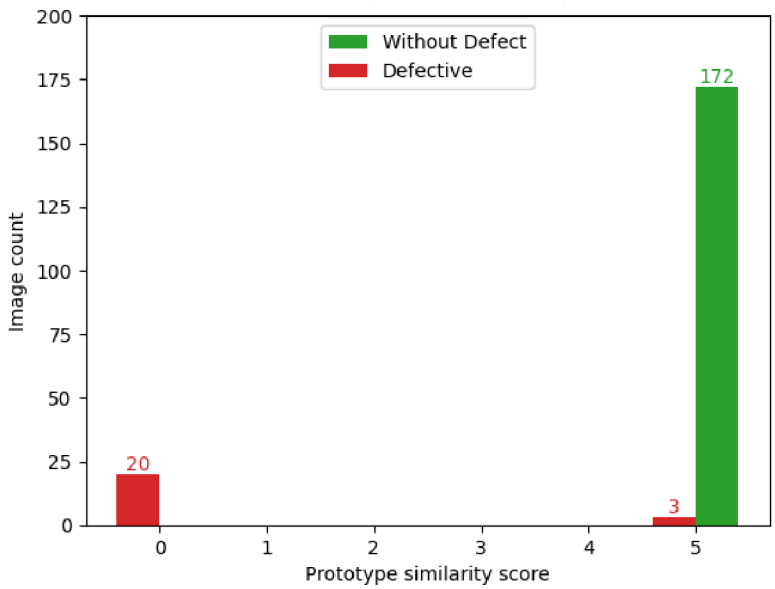


Figure 5.9: The performance of the base architecture, trained on SE images for 60 epochs, with lowered dropout to 0.2. Its BSE counterpart scored 23 FN. Each column represents the number of samples similar to the number of prototypes given by the prototype similarity score. Score of 5 is desired for non-defective apertures, 0 for defective.

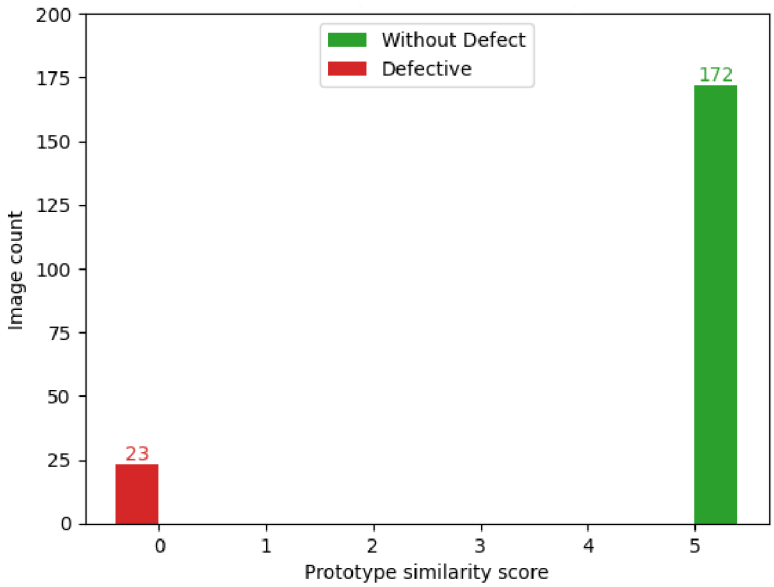


Figure 5.10: The performance of the best found model, using the base architecture without dropout, trained on SE images for 30 epochs.

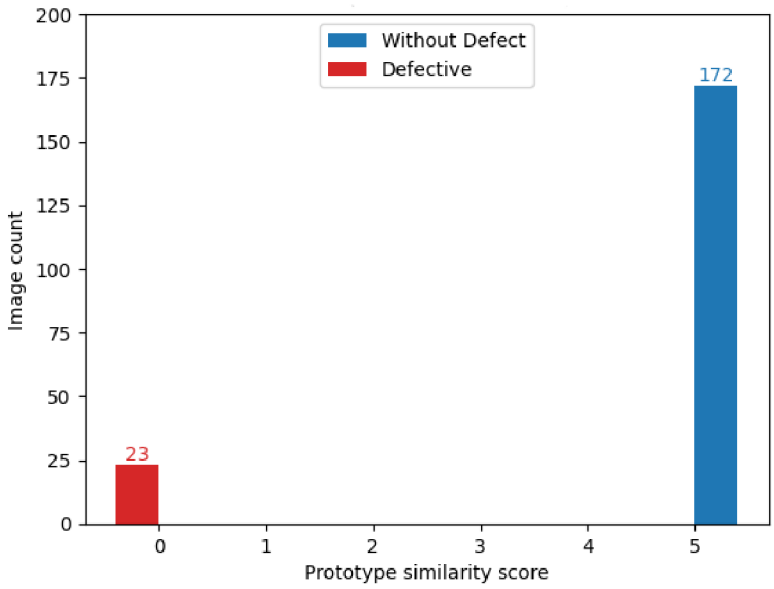


Figure 5.11: The performance of the best found model, using the base architecture without dropout, trained on BSE images for 30 epochs.

The effect of dropout on the performance of the models was so significant, that almost any other hyperparameter and architecture modifications still led to perfect anomaly detection capabilities, as long as there was no dropout present in the branches. Notable exception was making the branches too shallow, which degraded the performance of BSE-trained models, while not affecting their SE-trained counterparts. This hints at SE images being again easier to work with for the task of anomaly detection, as in the previous approaches. Other than this, learning rate increased to 10^{-3} or decreased to 10^{-5}, deeper branches, higher or lower number of filters in Conv2D layers and neurons in Dense layers, all led to the same results on the testing set, while only affecting the training time and learned margin between normal and anomalous samples.

Indeed, most of the examined models learned even higher margin between defective and non-defective samples than the enforced margin of 10 by the employed triplet loss. Enforcing lower margin of 1 was tested too, but was shown to be vastly inferior in the obtained results. Therefore a conclusion is reached that using larger triplet loss margins can lead to better anomaly detection performance.

Triplet networks converge faster than their Siamese counterparts, requiring as little as 20 training epochs to learn a decent enough margin between defective and non-defective samples. This translates to about 3 hours of training time for the base architecture. The training is also rather stable when dropout is removed, as repeated training runs produce the same results on the testing set. Interestingly, simpler convolutional layers performed better than the stacked variant in this approach, which translated to a slight decrease in detection time, ending up at about three quarters of a second per queried sample.

As the training data set of triplets was created by random selection, three such data sets were tested to see if the randomness affected the detection performance. Models capable of learning to detect anomalies in optimal fashion were able to do so no matter the training set used, and worse models were not found to perform better when trained on any of the three sets. Therefore in this scenario there is no need to employ techniques such as online triplet mining to boost the performance of already very fast convergence.

Triplet network approach confirmed its advantages over the Siamese network approach, providing faster convergence and even faster detection time. Triplet networks were also more lenient to architecture and hyperparameter choices, as long as dropout is omitted. This is an important advantage, as precise tuning of the model can take substantial time.

This approach further compounds that DDML models can be used successfully for the task of anomaly detection, as long as a handful of defective samples are available for training.

Again, the SE image type was shown to be easier for the models to perform on, but in this case, even on BSE images the perfect anomaly detection performance was reached quite easily. Therefore, as with Siamese networks, BSE and SE-trained model can be used in tandem, with samples with differing verdicts being left for further manual inspection.

5.4 Hybrid oc-SVM approach

As a host of models from previous approaches were tested as feature extractors, each with specific hyperparameter and architectural choices, for brevity, they will be referred to by their abbreviations introduced in the Appendix D.

As there are only two hyperparameters that control the oc-SVM behaviour, tuning them was a task for a simple grid search. Between the two γ options, used by the `scikit-learn` oc-SVM implementation, value *auto* was shown to be the better choice for most of the Siamese and Triplet branches used as feature extractors, while the option *scale* universally improved the performance of encoders by eliminating several false positives and negatives in comparison to *auto*. The fraction of allowed training false positives, controlled by v, was shown to be the best when set between 0.1 and 0.01, with the optimal value differing between models.

The downside of v is that in a semi-supervised setting, like in this approach, it cannot be set to 0 and the oc-SVM takes the liberty of flagging up to the allowed fraction of training samples as false positives, even if they are non-defective. This led to an observation that the best performing standalone models usually receive a big hit to

their performance when used in hybrid approach. The oc-SVM either does not learn a reasonable classification at all, or adds a few false positives to otherwise perfect anomaly detectors.

On the other hand, some of the models that performed poorly by themselves did drastically improve when used in hybrid approach. This was the case especially with encoders, where tested models **E1** and **E5** matched the BSE result of the best autoencoder model enhanced by granulometry and improved the best reached SE result to only 2 false positives, respectively.

Table 5.1: The best performing feature extractors of each approach on BSE-type images, together with their optimal oc-SVM hyperparameter settings and detection time per one sample.

Model	**FP**	**FN**	v	γ	**Detection time**
E1	9	3	0.05	scale	0.18 s
S1	7	0	0.02	auto	0.60 s
T5	1	0	0.01	auto	0.24 s

Table 5.2: The best performing feature extractors of each approach on SE-type images, together with their optimal oc-SVM hyperparameter settings and detection time per one sample.

Model	**FP**	**FN**	v	γ	**Detection time**
E5	2	0	0.01	scale	0.18 s
S3	10	0	0.05	auto	0.65 s
T1	0	0	0.02	scale	0.21 s

The model **S1** by itself has not learned to detect anomalies at all, yet combined with oc-SVM, immense improvement was reached. In contrast, the perfect Siamese and Triplet detectors have not worked well at all when used in hybrid model, usually producing upwards of 20 more false positives than when used standalone.

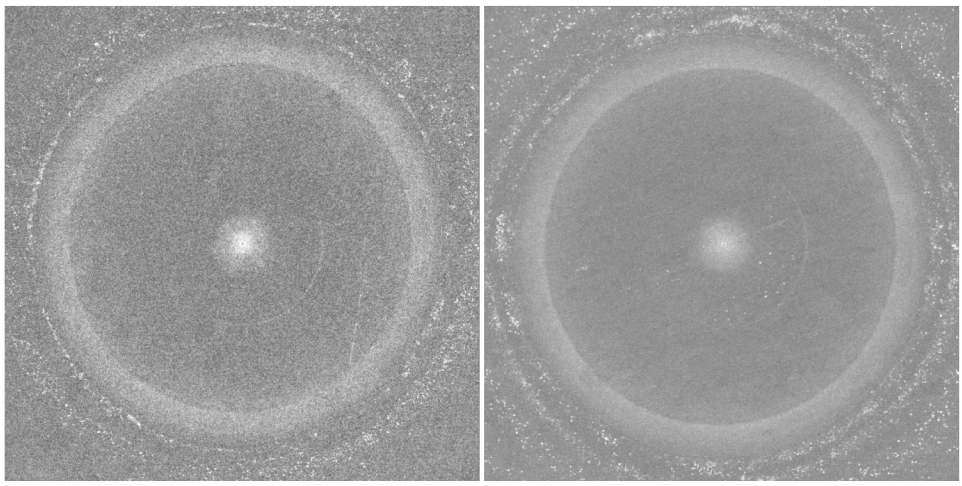


Figure 5.12: The two false positives of the **E5** model. Both images show heavy staining around the disk, which is usually acceptable.

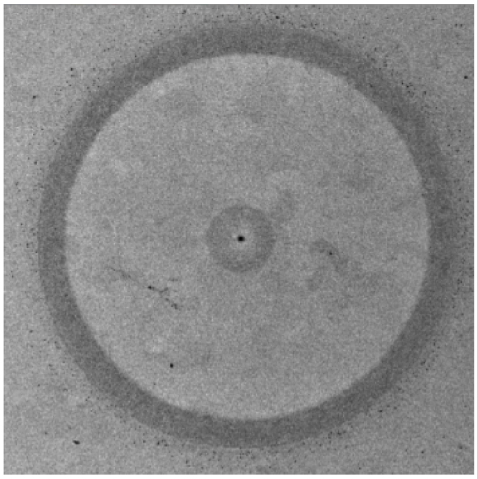


Figure 5.13: The single false positive of the **T5** model, with stains on the verge of acceptability.

One feature extractor managed to reach perfect anomaly detection on SE images and again it was the case of a worse standalone model being improved by hybrid approach. The model **T1** in combination with classification provided by oc-SVM eliminated 3 false negatives

it scored when used by itself, and also reduced the detection time by over three times, to one fifth of a second.

Two other models came close, with **T5** producing 1 false positive on BSE images and **E5** with 2 false positives on SE data. While not perfect, these are still commendable results. With the **T5** model, there is still the odd chance that perhaps the problematic image was misclassified when labeled by the expert, as it does show stains that could be on another day classified as defective.

Despite not being theoretically optimal, the hybrid approach produced competitive results to the previous approaches. It improved the anomaly detection ability of autoencoder architectures as expected, and in one case managed to achieve perfect anomaly detection on SE-type images using a branch of a trained Triplet network as the feature extractor. The latter case also provided significant detection time speed-up.

However, the hyperparameter v could not be set in a way that would preserve the optimal performance of other Siamese and Triplet networks, instead introducing training set false positives. Hence, with the exception of **T1**, a speed-up of such perfect detectors could not be achieved while maintaining their abilities.

Counter-intuitively, choosing the best standalone models proved futile, and by themselves unusable architectures showed great results when paired with the oc-SVM classifier. In this sense, the choice of a suitable feature extractor is surprisingly nuanced. When vying for the best possible anomaly detection performance using this approach, one should take care to test a plethora of differing models as feature extractors, as well as a wide range of v values.

5.5 Variational autoencoder approach

In this approach, the base architecture already managed to reach perfect anomaly detection of 0 false positives and negatives on SE images, while matching the best convolutional autoencoder model on BSE images with 6 false negatives and several false positives. However, the optimal decision threshold between defective and non-defective apertures had to be found manually, as the estimated one allowed a

number of false positives. The margin between reconstruction errors of acceptable apertures and defective ones is also very thin, hence further hyperparameter tuning was in order.

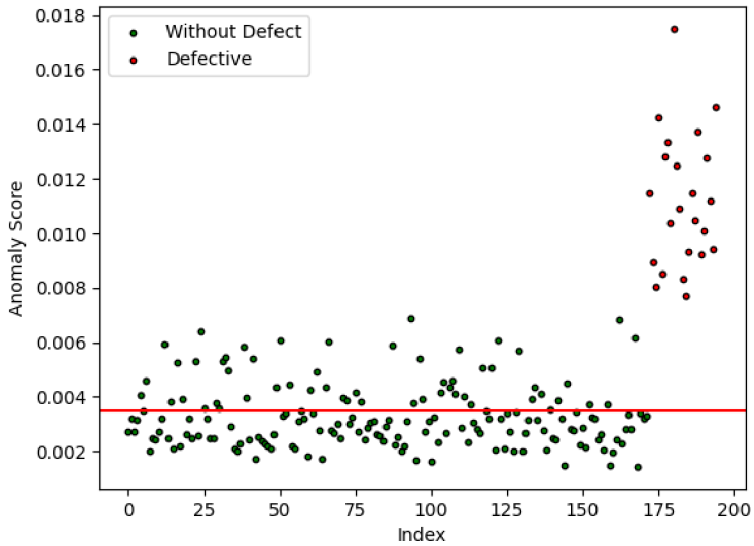


Figure 5.14: The base architecture's performance on SE-type images, trained for 800 epochs. Each dot is the achieved reconstruction error of a tested sample. The estimated decision threshold (the red line) in this case isn't that great, and a better one has to be found manually. While perfect anomaly detection is attainable with a careful choice, the margin between defective and acceptable samples is very slim.

The two main factors whose tuning led to improvement were the reconstruction loss factor α_{rl} and the latent space dimension. Lowering the latent dimension to just 6 and increasing α_{rl} to 10^4 each improved the BSE-trained models by shaving off a few false positives and achieved perfect anomaly detection on SE images with slightly wider margin. When combined together, these two changes produced the best obtained results on both BSE and SE images, reaching 20 FP with 2 FN and perfect anomaly detection with wide enough margin, respectively.

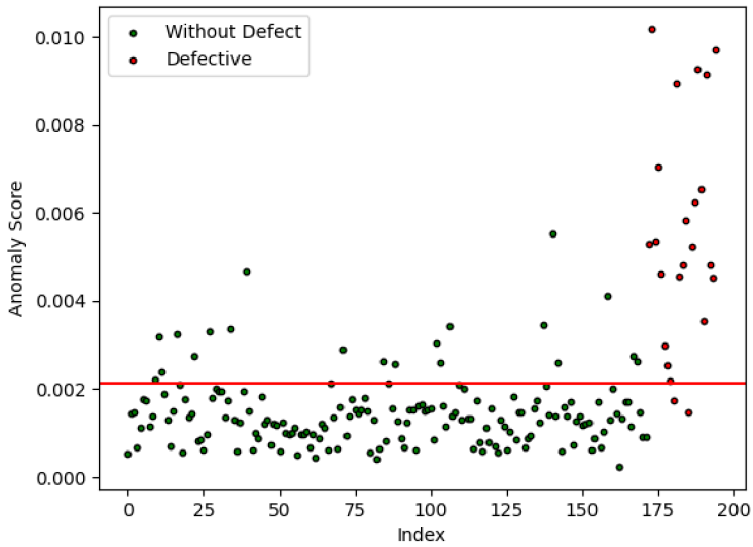


Figure 5.15: The performance of the best found model architecture for BSE images, trained for 1200 epochs. This time, the decision threshold estimate is basically as good as it gets.

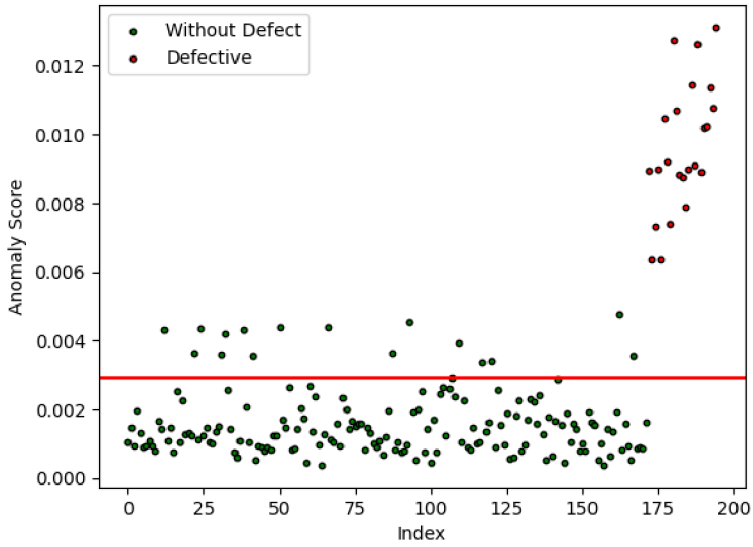


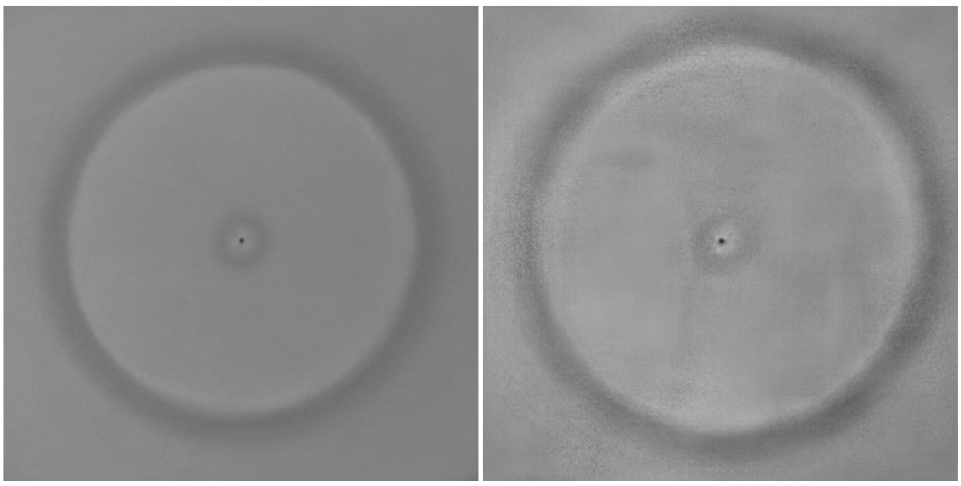
Figure 5.16: The performance of the best found model architecture for SE images, trained for 800 epochs. Acceptable value for decision threshold lies between 0.005 and 0.006.

Increasing the the reconstruction loss factor even further still produces optimal anomaly detection performance on SE images, albeit with thinner margin again, but increases the FN count on BSE images. No hyperparameter and architecture choices could be found that would achieve perfect anomaly detection results on BSE images, as this approach once again proves the SE-type images to be easier to work with for the task of anomaly detection. With BSE images, only trade-offs between false positives and false negatives can be obtained by moving the decision threshold.

As for other hyperparameter choices, changing the learning rate in any way, increasing or decreasing the number of filters in Conv2D layers, or deepening the encoder and decoder networks all led to degradation of performance. Increasing the latent space dimension also proved counterproductive. Conv2DTranspose layers were again tested in the decoder network, and again proved by far the inferior choice, unable to even come close to perfect anomaly detection on either of the image types.

Training of the models was very stable, much like with the convolutional autoencoders, producing almost identical results across multiple training runs. The optimal number of training epochs was shown to be between 800 and 1200 for most of the models. As the lower resolution images were used, the training itself was very swift, taking just under 2 hours. The best found models have also fairly low prediction time, requiring 0.27 s per queried sample.

The generated images were sharper than in the case of the convolutional autoencoders, and even with the base latent dimension of 12 the models did not learn to reconstruct stains. Instead, image regions containing stains were noticeably blurrier, contributing to higher reconstruction loss. This is exactly the desired behaviour.



(a) Non-defective aperture. (b) Aperture with stains.

Figure 5.17: Reconstructions of two apertures, with visible blur on the left side of the defective aperture, where a bunch of stains occur.

Using variational autoencoders led to optimal performance on at least the SE images. However, the decision threshold had to be manually corrected. In scenario where no defective samples would be available for testing and optimizing the decision threshold, setting it slightly above the maximum reconstruction error obtained on the training data set could also work well. On BSE-type images, the performance reached rivals that of the best autoencoder model enhanced with granulometry, while being overshadowed by the DDML approaches or the best found hybrid models.

The prediction speed of trained VAE models is excellent. The only downside is that the variational loss, due to its complexity, cannot be saved as a part of the architecture, unlike in the previous approaches. Therefore, the potential anomaly detector has to include the model and the loss classes, reconstruct the model, and finally load the weights obtained by training.

5.6 BiGAN approach

The base BiGAN architecture showed performance resembling that of the first convolutional autoencoder efforts. Not great by any means on either of the image types, but showing signs that perhaps careful tuning could bear significant improvement.

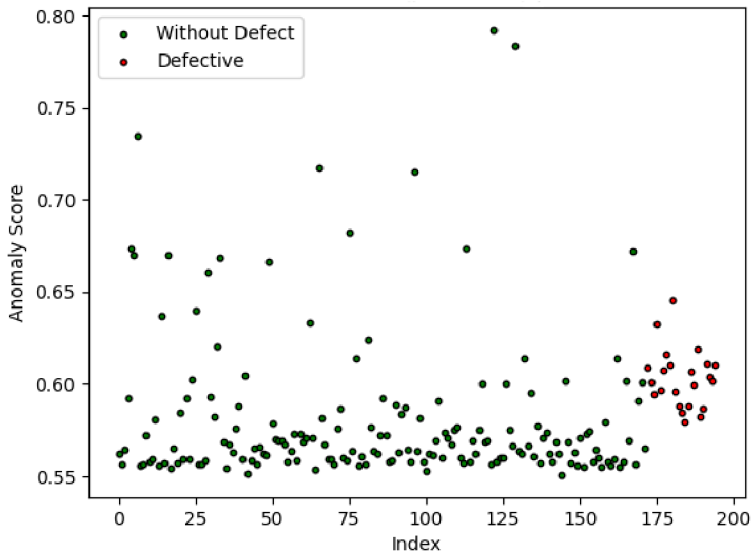


Figure 5.18: The base model trained for 1000 epochs on SE-type images, using binary cross-entropy as the loss function. Each dot represents the anomaly score of a single image, obtained as the sum of reconstruction and discriminator losses. Decision threshold estimates were omitted in this approach, as early on during testing it was shown that they worked extremely poorly.

However, despite investing more time into architecture and hyperparameter tuning than in any other of the approaches, the tested BiGAN models lived up to the infamous reputation of GANs for being extremely difficult to get good results from.

Interestingly, binary cross-entropy loss vastly outperformed the Wasserstein loss in this scenario, as models using the Wasserstein loss all failed due to the loss oscillating wildly or, more frequently, due to mode collapse. Employing techniques such as label smoothing,

using AveragePooling2D layers instead of MaxPooling2D layers, extra discriminator or encoder and generator training steps, fine-tuning the weight clipping or removing it altogether, employing different optimizers, nothing managed to solve these problems. More traditional changes such as increasing or decreasing the learning rate, the number of filters and neurons, or the depth of the three neural networks were also in vain. And as models using the Wasserstein loss were unable to even learn to reconstruct the samples, their anomaly detection performance was extremely poor.

In contrast, using the binary cross-entropy as the loss function led to fairly stable training, and with some modifications to the base architecture even generative capabilities were attained, without mode collapse. The best found models for both image types were obtained by adding batch normalization to the encoder and generator networks. Additionally, the best performing BSE-trained model was 2 layers shallower in both encoder and generator network. Traditional neural network hyperparameter tuning or having either of the sub-networks train for more steps per each epoch led either to no discernable changes in performance from the base architecture, or to noticeable deterioration. Surprisingly, even changing the latent space dimension did not produce better results.

Even then, the best attained models performed slightly worse than the best ones from the baseline convolutional autoencoder approach, for each image type. Testing even lower resolution images (192×192) also proved futile, as even worse anomaly detection abilities were displayed by the tested models.

Training itself was quite fast, with 1000 epochs taking under 2 hours for most of the tested models. Optimal number of training epochs varied wildly, with the best found model for SE images requiring only 500 epochs, while some models needed as much as 8000 epochs to reach their best anomaly detection performance. Training the same model several times also led to sometimes strikingly different results, as even the most stable models could not match the stability of the previous approaches. And as all three networks, which comprise the BiGAN model, are used in the detection process itself, the time of 0.55 s per queried sample is not among the best.

The anomaly detection score was largely dependent on the reconstruction error, as the discriminator networks of the tested models

were unable to reach good enough classification skills to discern between defective and non-defective samples. This was mainly due to the inability of the generator to produce quality images upon which the discriminator could be properly challenged. Hence, tuning the weights of each of the two terms of the anomaly score mattered little, as the discriminator loss was almost identical for all of the samples.

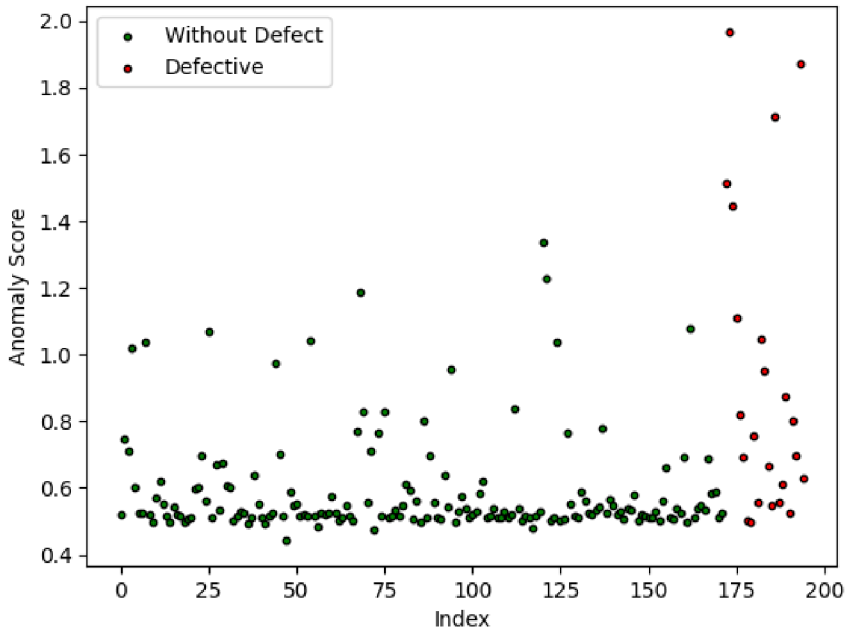


Figure 5.19: The best performing model on BSE-type images, slightly shallower and with extra batch normalization, trained for 8000 epochs with cross entropy as the loss function. With the decision threshold set to 0.57, 6 false negatives with 36 false positives are reachable. Other models reached the same number of false positives with far more false negatives.

With the best BiGAN models outperformed even by the convolutional autoencoders, the findings of this approach serve as a reminder that the "hottest" new techniques might not always be the best choice available, even in the task of anomaly detection. The low number of training samples, coupled with their rather high dimensionality, were probably the two main factors behind the poor performance. After all,

adapting GANs to larger resolutions is still an ongoing research topic and a low number of training data is the bane of many neural network models.

The best obtained performance on the SE-type images is still fairly decent, with majority of the false positives again being samples with stains on the verge of unacceptable. This approach further confirms that the SE images are better suited for any anomaly detection purposes.

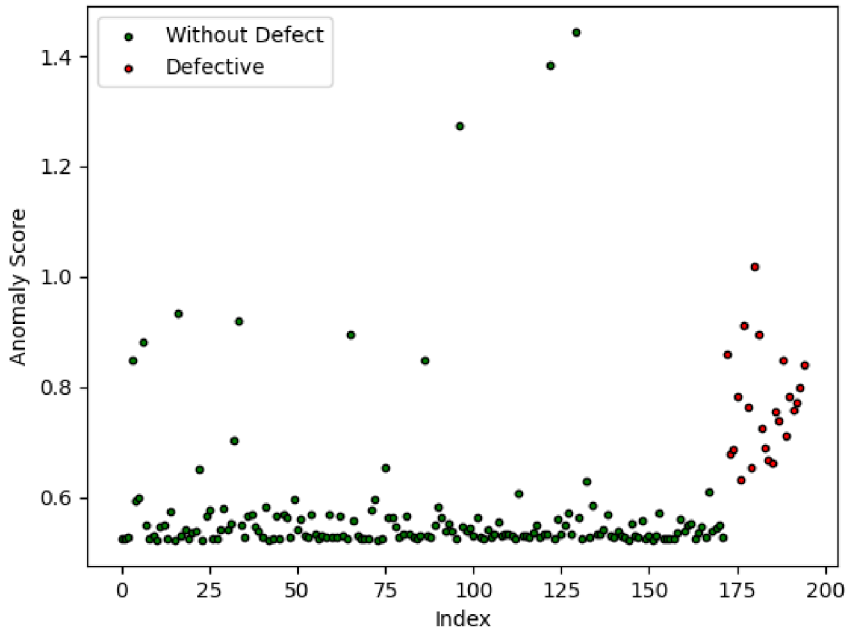


Figure 5.20: The best performing model on SE-type images, with extra batch normalization, trained for 500 epochs with cross entropy as the loss function. With the decision threshold set to 0.62, 0 false negatives with 13 false positives can be reached, which is slightly worse than the best convolutional autoencoder model. This best found model still vastly outperforms all of the other tested BiGAN models.

6 Conclusion

A thorough overview of neural network architectures used for the task of anomaly detection on image data, divided into categories by their principles, was provided in the Chapter 2 of this thesis. Out of these, six approaches were chosen to be tested as potential defect detectors on the provided data set. This data set consists of two types of images of electron-optical apertures, both represented equally by 172 non-defective and 23 defective samples. The following tables present the performances of the best found models per each approach, on both of the image types, coupled with their detection speed. For immediate clarity, the performance is measured as the number of false positives (non-defective samples marked as anomalous) and false negatives (defective samples marked as acceptable).

Table 6.1: The best results achieved on BSE-type images.

Approach	**False positives**	**False negatives**	**Detection**
Autoencoder	9	3	0.35 s
Siamese network	0	0	0.97 s
Triplet network	0	0	0.76 s
Hybrid	1	0	0.24 s
Variational AE	20	2	0.27 s
BiGAN	36	6	0.55 s

Table 6.2: The best results achieved on SE-type images.

Approach	**False positives**	**False negatives**	**Detection**
Autoencoder	10	0	0.33 s
Siamese network	0	0	0.97 s
Triplet network	0	0	0.76 s
Hybrid	0	0	0.21 s
Variational AE	0	0	0.27 s
BiGAN	13	0	0.59 s

If the provided data set contains correctly labeled samples, four of the tested approaches show the ability to provide perfect defect detection in some capacity. Notably, the SE-type images proved to be much better suited for the anomaly detection task then their BSE counterparts.

It's worth noting that if no defective samples were available for training, only the variational autoencoder approach would reach the optimal detection abilities, and only on the SE-type images. On the other hand, even with very few defective samples available to train them, the Siamese and Triplet networks showcased outstanding performance, being the only approaches to work perfectly even on the BSE images. Their only downside being the detection speed per sample. However, with the number of tested apertures being in the hundreds per year, detection speed might not be that much of a concern in this scenario.

Traditional convolutional autoencoders struggled with images showing stains on the verge between acceptable and defective, but could perhaps be used to highlight problematic images to perform a double-check of the assigned labels by the human expert. Despite the emerging success of GANs in anomaly detection, the tested BiGAN architecture suffered from not enough training samples and the necessity for the images to remain at high resolution. Hybrid models proved again that despite not being theoretically optimal, they can provide extremely good performance and are very easy to employ. In fact, hybrid model with a branch from Triplet network serving as the feature extractor reached the optimal performance with the fastest detection time.

However, as the detection time might not be such a big concern due to the low amount of apertures manufactured yearly, the best possible automated defect detector could in fact be an ensemble of all of the models that reached the optimal performance. This way, each model would cast its prediction and if the result wasn't unanimous, the aperture would be left for the human expert to decide. This could lead to a very robust system that would reliably save time of the human expert. Of course, as both image types are captured for each aperture, using any of the optimal models by themselves is an option too.

Although using neural networks as automated defect detectors for this specific manufacturing process shows great promise, the ex-

tremely low amount of defective samples for testing purposes tempers the optimism. Of special concern should be apertures with their central hole filled with debris, as only 3 such images are provided in the data set, and each of them contains stains that could by themselves lead to the sample being classified as defective. If such plugged apertures would occur without any other defects, the trained models might not be able to correctly detect the defect. As for the other defect types, the best found models showed that they can handle those reliably.

In conclusion, several approaches of using neural network architecture for the purpose of defect detection reached perfect detection capabilities on the provided data set, and hint that their usage in practice could very much be possible. Nonetheless, further testing on additional samples, especially defective ones, would be desirable to gain better confidence.

Bibliography

1. CHANDOLA, Varun; BANERJEE, Arindam; KUMAR, Vipin. Anomaly Detection: A Survey. *ACM Comput. Surv.* 2009, vol. 41. Available from DOI: 10.1145/1541880.1541882.
2. PIMENTEL, Marco A.F.; CLIFTON, David A.; CLIFTON, Lei; TARASSENKO, Lionel. A review of novelty detection. *Signal Processing*. 2014, vol. 99, pp. 215–249. ISSN 0165-1684. Available from DOI: https://doi.org/10.1016/j.sigpro.2013.12.026.
3. VEERAMREDDY, Jyothsna; PRASAD, V.; PRASAD, Koneti. A Review of Anomaly based Intrusion Detection Systems. *International Journal of Computer Applications*. 2011, vol. 28, pp. 26–35. Available from DOI: 10.5120/3399-4730.
4. PATCHA, Animesh; PARK, Jung-Min. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*. 2007, vol. 51, no. 12, pp. 3448–3470. ISSN 1389-1286. Available from DOI: https://doi.org/10.1016/j.comnet.2007.02.001.
5. ADEWUMI, Aderemi O.; AKINYELU, Andronicus A. A survey of machine-learning and nature-inspired based credit card fraud detection techniques. *International Journal of System Assurance Engineering and Management*. 2017, vol. 8, no. 2, pp. 937–953. Available from DOI: 10.1007/s13198-016-0551-y.
6. KIRAN, B. Ravi; THOMAS, Dilip Mathew; PARAKKAL, Ranjith. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *CoRR*. 2018, vol. abs/1801.03149. Available from arXiv: 1801.03149.
7. DIEHL, Chris; HAMPSHIRE, John. Real-time object classification and novelty detection for collaborative video surveillance. In: 2002, vol. 3, pp. 2620–2625. ISBN 0-7803-7278-6. Available from DOI: 10.1109/IJCNN.2002.1007557.
8. MARKOU, Markos; SINGH, Sameer. A Neural Network-Based Novelty Detector for Image Sequence Analysis. *IEEE transactions on pattern analysis and machine intelligence*. 2006, vol. 28, pp. 1664–77. Available from DOI: 10.1109/TPAMI.2006.196.

9. BALL, John E.; ANDERSON, Derek T.; CHAN, Chee Seng. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*. 2017, vol. 11, no. 04, p. 1. ISSN 1931-3195. Available from DOI: 10.1117/1.jrs.11.042609.

10. ZHANG, Yang; MERATNIA, Nirvana; HAVINGA, Paul. Outlier Detection Techniques for Wireless Sensor Networks: A Survey. *Communications Surveys & Tutorials, IEEE*. 2010, vol. 12, pp. 159–170. Available from DOI: 10.1109/SURV.2010.021510.00088.

11. KWON, Donghwoon; KIM, Hyunjoo; KIM, Jinoh; SUH, Sang; KIM, Ikkyun; KIM, Kuinam. A survey of deep learning-based network anomaly detection. *Cluster Computing*. 2019, vol. 22. Available from DOI: 10.1007/s10586-017-1117-8.

12. MIN, Erxue; LONG, J.; LIU, Q.; CUI, Jianjing; CAI, Zhiping; MA, Junbo. SU-IDS: A Semi-supervised and Unsupervised Framework for Network Intrusion Detection. In: *ICCCS*. 2018.

13. LIN, Zilong; SHI, Yong; XUE, Zhi. *IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection*. 2019. Available from arXiv: 1809.02077 [cs.CR].

14. LITJENS, Geert; KOOI, Thijs; BEJNORDI, Babak Ehteshami; SETIO, Arnaud Arindra Adiyoso; CIOMPI, Francesco; GHAFOORIAN, Mohsen; LAAK, Jeroen A.W.M. van der; GINNEKEN, Bram van; SÁNCHEZ, Clara I. A survey on deep learning in medical image analysis. *Medical Image Analysis*. 2017, vol. 42, pp. 60–88. ISSN 1361-8415. Available from DOI: 10.1016/j.media.2017.07.005.

15. ALAVERDYAN, Zaruhi. *Unsupervised representation learning for anomaly detection on neuroimaging. Application to epilepsy lesion detection on brain MRI*. 2019. PhD thesis. Université de Lyon.

16. CLIFTON, Lei; CLIFTON, David; WATKINSON, Peter; TARASSENKO, L. Identification of Patient Deterioration in Vital-Sign Data using One-Class Support Vector Machines. In: 2011, pp. 125–131.

17. QUINN, John; WILLIAMS, Christopher. Known Unknowns: Novelty Detection in Condition Monitoring. In: 2007, vol. 4477, pp. 1–6. ISBN 978-3-540-72846-7. Available from DOI: 10.1007/978-3-540-72847-4_1.

18. TARASSENKO, L.; HAYTON, P.; CERNEAZ, N.; BRADY, M. Novelty detection for the identification of masses in mammograms. In: 1995.

19. GROSJEAN, Bénédicte; MOISAN, Lionel. A-contrario Detectability of Spots in Textured Backgrounds. *Journal of Mathematical Imaging and Vision*. 2009, vol. 33. Available from DOI: 10.1007/s10851-008-0111-4.

20. ZONTAK, Maria; COHEN, I. Defect detection in patterned wafers using anisotropic kernels. *Machine Vision and Applications*. 2008, vol. 21, pp. 129–141.

21. TOUT, Karim; RETRAINT, F.; COGRANNE, R. Automatic Vision System for Wheel Surface Inspection and Monitoring. In: 2017.

22. LI, Wei-Chen; TSAI, Du-Ming. Wavelet-Based Defect Detection in Solar Wafer Images with Inhomogeneous Texture. *Pattern Recogn*. 2012, vol. 45, no. 2, pp. 742–756. ISSN 0031-3203. Available from DOI: 10.1016/j.patcog.2011.07.025.

23. NEOGI, N.; MOHANTA, D.; DUTTA, P. K. Review of vision-based steel surface inspection systems. *EURASIP Journal on Image and Video Processing*. 2014, vol. 2014, pp. 1–19.

24. XIE, Xianghua. A Review of Recent Advances in Surface Defect Detection using Texture analysis Techniques. *ELCVIA. Electronic letters on computer vision and image analysis, ISSN 1577-5097, Vol. 7, Nº. 3, 2008* (*Ejemplar dedicado a: Quality Control by Artificial Vision*)*, pags. 11-22*. 2008, vol. 7.

25. STAAR, Benjamin; LÜTJEN, Michael; FREITAG, Michael. Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*. 2019, vol. 79, pp. 484–489. ISSN 2212-8271. Available from DOI: https://doi.org/10.1016/j.procir.2019.02.123. 12th CIRP Conference on Intelligent

Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy.

26. PITTINO, Federico; PUGGL, Michael; MOLDASCHL, Thomas; HIRSCHL, Christina. Automatic Anomaly Detection on In-Production Manufacturing Machines Using Statistical Learning Methods. *Sensors*. 2020, vol. 20, no. 8. ISSN 1424-8220. Available from DOI: 10.3390/s20082344.

27. MASCI, J.; MEIER, U.; CIRESAN, Dan C.; SCHMIDHUBER, J.; FRICOUT, G. Steel defect classification with Max-Pooling Convolutional Neural Networks. *The 2012 International Joint Conference on Neural Networks (IJCNN)*. 2012, pp. 1–6.

28. LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. *Fully Convolutional Networks for Semantic Segmentation*. 2015. Available from arXiv: 1411.4038 [cs.CV].

29. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition*. 2015. Available from arXiv: 1512.03385 [cs.CV].

30. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*. 2012, vol. 25. Available from DOI: 10.1145/3065386.

31. CHALAPATHY, Raghavendra; CHAWLA, Sanjay. Deep Learning for Anomaly Detection: A Survey. *CoRR*. 2019, vol. abs/1901.03407. Available from arXiv: 1901.03407.

32. JAVAID, Ahmad; NIYAZ, Quamar; SUN, Weiqing; ALAM, Mansoor. A Deep Learning Approach for Network Intrusion Detection System. In: 2015, vol. 3. Available from DOI: 10.4108/eai.3-12-2015.2262516.

33. PENG, Huan-Kai; MARCULESCU, Radu. Multi-Scale Compositionality: Identifying the Compositional Structures of Social Dynamics Using Deep Learning. *PloS one*. 2015, vol. 10, e0118309. Available from DOI: 10.1371/journal.pone.0118309.

34. HAWKINS, D. M. *Identification of outliers*. London [u.a.]: Chapman and Hall, 1980. Monographs on applied probability and statistics. ISBN 041221900X. Available also from: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+02435757X&sourceid=fbw_bibsonomy.

35. MARKOU, Markos; SINGH, Manjeet. Novelty detection: A review—Part 1: Statistical Approaches. *Signal Processing*. 2003, vol. 83, pp. 2481–2497. Available from DOI: 10.1016/j.sigpro.2003.07.018.

36. ANDREWS, Jerone Theodore Alexander. *Representation learning for anomaly detection in computer vision*. 2018. PhD thesis. University College London.

37. GOERNITZ, N.; KLOFT, M.; RIECK, K.; BREFELD, U. Toward Supervised Anomaly Detection. *Journal of Artificial Intelligence Research*. 2013, vol. 46, pp. 235–262. ISSN 1076-9757. Available from DOI: 10.1613/jair.3623.

38. EHRET, Thibaud; DAVY, Axel; MOREL, Jean-Michel; DELBRACIO, Mauricio. Image Anomalies: A Review and Synthesis of Detection Methods. *Journal of Mathematical Imaging and Vision*. 2019, vol. 61, no. 5, pp. 710–743. ISSN 1573-7683. Available from DOI: 10.1007/s10851-019-00885-0.

39. RUFF, Lukas; VANDERMEULEN, Robert; GOERNITZ, Nico; DEECKE, Lucas; SIDDIQUI, Shoaib Ahmed; BINDER, Alexander; MÜLLER, Emmanuel; KLOFT, Marius. Deep One-Class Classification. In: DY, Jennifer; KRAUSE, Andreas (eds.). *Proceedings of the 35th International Conference on Machine Learning*. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, vol. 80, pp. 4393–4402. Proceedings of Machine Learning Research. Available also from: http://proceedings.mlr.press/v80/ruff18a.html.

40. LECUN, Yann; BENGIO, Y.; HINTON, Geoffrey. Deep Learning. *Nature*. 2015, vol. 521, pp. 436–44. Available from DOI: 10.1038/nature14539.

41. COLLOBERT, Ronan; WESTON, Jason; BOTTOU, Leon; KARLEN, Michael; KAVUKCUOGLU, Koray; KUKSA, Pavel. *Natural Language Processing (almost) from Scratch*. 2011. Available from arXiv: 1103.0398 [cs.LG].

42. HINTON, G.; DENG, L.; YU, D.; DAHL, G. E.; MOHAMED, A.; JAITLY, N.; SENIOR, A.; VANHOUCKE, V.; NGUYEN, P.; SAINATH, T. N.; KINGSBURY, B. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*. 2012, vol. 29, no. 6, pp. 82–97. Available from DOI: 10.1109/MSP.2012.2205597.

43. WALSH, Joseph; O' MAHONY, Niall; CAMPBELL, Sean; CARVALHO, Anderson; KRPALKOVA, Lenka; VELASCO-HERNANDEZ, Gustavo; HARAPANAHALLI, Suman; RIORDAN, Daniel. Deep Learning vs. Traditional Computer Vision. In: 2019. ISBN 978-981-13-6209-5. Available from DOI: 10.1007/978-3-030-17795-9_10.

44. MASANA, Marc; RUIZ, Idoia; SERRAT, Joan; WEIJER, Joost van de; LOPEZ, Antonio M. *Metric Learning for Novelty and Anomaly Detection*. 2018. Available from arXiv: 1808.05492 [cs.CV].

45. AN, Jinwon; CHO, S. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. In: 2015.

46. XIA, Yan; CAO, Xudong; WEN, Fang; HUA, Gang; SUN, Jian. Learning Discriminative Reconstructions for Unsupervised Outlier Removal. In: 2015, pp. 1511–1519. Available from DOI: 10.1109/ICCV.2015.177.

47. RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Internal Representations by Error Propagation. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN 026268053X.

48. HINTON, G.E.; SALAKHUTDINOV, R.R. Reducing the Dimensionality of Data with Neural Networks. *Science (New York, N.Y.)* 2006, vol. 313, pp. 504–7. Available from DOI: 10.1126/science.1127647.

49. WENG, Lilian. *From Autoencoder to Beta-VAE* [online]. 2018 [visited on 2020-12-04]. Available from: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html.

50. SHAHIN SHAMSABADI, Ali; BABAIE-ZADEH, Massoud; SEYYEDSALEHI, Seyyede; RABIEE, Hamid; JUTTEN, Christian. A New Algorithm for Training Sparse Autoencoders. In: 2017. Available from DOI: 10.23919/EUSIPCO.2017.8081588.

51. RIFAI, Salah; VINCENT, Pascal; MULLER, Xavier; GLOROT, Xavier; BENGIO, Y. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In: 2011.

52. VINCENT, Pascal; LAROCHELLE, Hugo; BENGIO, Y.; MANZAGOL, Pierre-Antoine. Extracting and composing robust features with denoising autoencoders. In: 2008, pp. 1096–1103. Available from DOI: 10.1145/1390156.1390294.

53. SATO, Daisuke; HANAOKA, Shouhei; NOMURA, Yukihiro; TAKENAGA, Tomomi; MIKI, Soichiro; YOSHIKAWA, Takeharu; HAYASHI, Naoto; ABE, Osamu. A primitive study on unsupervised anomaly detection with an autoencoder in emergency head CT volumes. In: 2018, p. 60. Available from DOI: 10.1117/12.2292276.

54. COWTON, J.; KYRIAZAKIS, I.; PLÖTZ, T.; BACARDIT, J. A Combined Deep Learning GRU-Autoencoder for the Early Detection of Respiratory Disease in Pigs Using Multiple Environmental Sensors. *Sensors*. 2018, vol. 18, no. 8.

55. CASTELLINI, Jacopo; POGGIONI, Valentina; SORBI, Giulia. Fake Twitter followers detection by denoising autoencoder. In: 2017, pp. 195–202. Available from DOI: 10.1145/3106426.3106489.

56. DAI, J.; SONG, H.; SHENG, G.; JIANG, X. Cleaning Method for Status Monitoring Data of Power Equipment Based on Stacked Denoising Autoencoders. *IEEE Access*. 2017, vol. 5, pp. 22863–22870. Available from DOI: 10.1109/ACCESS.2017.2740968.

57. NOLLE, Timo; LUETTGEN, Stefan; SEELIGER, Alexander; MÜHLHÄUSER, Max. Analyzing business process anomalies using autoencoders. *Machine Learning*. 2018, vol. 107, no. 11, pp. 1875–1893. ISSN 1573-0565. Available from DOI: 10.1007/s10994-018-5702-8.

58. LUO, H.; ZHONG, S. Gas turbine engine gas path anomaly detection using deep learning with Gaussian distribution. In: *2017 Prognostics and System Health Management Conference (PHM-Harbin)*. 2017, pp. 1–6. Available from DOI: 10.1109/PHM.2017.8079166.

59. QU, Aaron; HE, Miao; DEUTSCH, Jason; HE, David. Detection of Pitting in Gears Using a Deep Sparse Autoencoder. *Applied Sciences*. 2017, vol. 7, p. 515. Available from DOI: 10.3390/app7050515.

60. YUAN, Y.; JIA, K. A Distributed Anomaly Detection Method of Operation Energy Consumption Using Smart Meter Data. In: *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*. 2015, pp. 310–313. Available from DOI: 10.1109/IIH-MSP.2015.38.

61. HASELMANN, Matthias; GRUBER, Dieter P.; TABATABAI, Paul. *Anomaly Detection using Deep Learning based Image Completion*. 2018. Available from arXiv: 1811.06861 [cs.CV].

62. YANG, Kun; ZHANG, Junjie; XU, Yang; CHAO, Jonathan. DDoS Attacks Detection with AutoEncoder. In: 2020, pp. 1–9. Available from DOI: 10.1109/NOMS47738.2020.9110372.

63. RIBEIRO, Manassés; LAZZARETTI, André Eugênio; LOPES, Heitor Silvério. A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*. 2018, vol. 105, pp. 13–22. ISSN 0167-8655. Available from DOI: https://doi.org/10.1016/j.patrec.2017.07.016. Machine Learning and Applications in Artificial Intelligence.

64. MUNAWAR, Asim; VINAYAVEKHIN, Phongtharin; MAGISTRIS, Giovanni. Limiting the Reconstruction Capability of Generative Neural Network using Negative Learning. In: 2017. Available from DOI: 10.1109/MLSP.2017.8168155.

65. RAO, Dattaraj; MITTAL, Shruti; RITIKA, S. Siamese Neural Networks for One-shot detection of Railway Track Switches. 2017.

66. KOCH, Gregory; ZEMEL, Richard; SALAKHUTDINOV, Ruslan. Siamese Neural Networks for One-shot Image Recognition. In: 2015.

67. VEAL, Charlie; SCHULZ, Jeffrey; BUCK, Andrew; ANDERSON, Derek; KELLER, James; POPESCU, Mihail; SCOTT, Grant; HO, Dominic; WILKIN, Timothy. Doing more with less: similarity neural nets and metrics for small class imbalanced data sets. In: BISHOP, Steven S.; ISAACS, Jason C. (eds.). *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV*. SPIE, 2020, vol. 11418, pp. 1–12. Available from DOI: 10.1117/12.2558092.

68. JMILA, Houda; IBN KHEDHER, Mohamed; BLANC, Gregory; EL YACOUBI, Mounim. Siamese Network Based Feature Learning for Improved Intrusion Detection. In: 2019, pp. 377–389. Available from DOI: 10.1007/978-3-030-36708-4_31.

69. BROMLEY, Jane; BENTZ, James; BOTTOU, Leon; GUYON, Isabelle; LECUN, Yann; MOORE, Cliff; SACKINGER, Eduard; SHAH, Rookpak. Signature Verification using a "Siamese" Time Delay Neural Network. *International Journal of Pattern Recognition and Artificial Intelligence*. 1993, vol. 7, p. 25. Available from DOI: 10.1142/S0218001493000339.

70. SUNDIN, Roberto Castro; RÖNNQVIST, Tony; GONZÁLEZ, Alejandro Sarmiento; WESTBERG, Simon. *Siamesifying the COVID-Net* [online]. 2020 [visited on 2020-12-05]. Available from: https://people.kth.se/~rosun/deep-learning/index.html.

71. HADSELL, Raia; CHOPRA, Sumit; LECUN, Yann. Dimensionality Reduction by Learning an Invariant Mapping. In: 2006, pp. 1735–1742. ISBN 0-7695-2597-0. Available from DOI: 10.1109/CVPR.2006.100.

72. HAN, Xufeng; LEUNG, Thomas; JIA, Y.; SUKTHANKAR, R.; BERG, A. MatchNet: Unifying feature and metric learning for patch-based matching. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3279–3286.

73. WANG, Jiang; SONG, Yang; LEUNG, Thomas; ROSENBERG, Chuck; WANG, Jinbin; PHILBIN, James; CHEN, Bo; WU, Ying. *Learning Fine-grained Image Similarity with Deep Ranking*. 2014. Available from arXiv: 1404.4661 [cs.CV].

74. MOINDROT, Olivier. *Triplet Loss and Online Triplet Mining in TensorFlow* [online]. 2018 [visited on 2020-12-06]. Available from: https://omoindrot.github.io/triplet-loss.

75. SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015. ISBN 9781467369640. Available from DOI: 10.1109/cvpr.2015.7298682.

76. SONG, Hyun Oh; XIANG, Yu; JEGELKA, Stefanie; SAVARESE, Silvio. *Deep Metric Learning via Lifted Structured Feature Embedding*. 2015. Available from arXiv: 1511.06452 [cs.CV].

77. HOFFER, Elad; AILON, Nir. *Deep metric learning using Triplet network*. 2018. Available from arXiv: 1412.6622 [cs.LG].

78. ROUSSI, M.; NORD, B. Implementation of Semi-Supervised Learning with Siamese Networks for Object Detection. In: *American Astronomical Society Meeting Abstracts #235*. 2020, vol. 235, p. 109.23. American Astronomical Society Meeting Abstracts.

79. UTKIN, L. V.; ZABOROVSKY, V. S.; LUKASHIN, A. A.; POPOV, S. G.; PODOLSKAJA, A. V. A Siamese Autoencoder Preserving Distances for Anomaly Detection in Multi-robot Systems. In: *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*. Los Alamitos, CA, USA: IEEE Computer Society, 2017, pp. 39–44. Available from DOI: 10.1109/ICCAIRO.2017.17.

80. RAMACHANDRA, Bharathkumar; JONES, Michael J.; VATSAVAI, Ranga Raju. *Learning a distance function with a Siamese network to localize anomalies in videos*. 2020. Available from arXiv: 2001.09189 [cs.CV].

81. WILKINS, Benedict; WATKINS, Chris; STATHIS, Kostas. *A Metric Learning Approach to Anomaly Detection in Video Games*. 2020. Available from arXiv: 2005.10211 [cs.LG].

82. TAYEH, Tareq; ABURAKHIA, Sulaiman; MYERS, Ryan; SHAMI, Abdallah. *Distance-Based Anomaly Detection for Industrial Surfaces Using Triplet Networks*. 2020. Available from arXiv: 2011.04121 [cs.CV].

83. DURKOTA, K.; LINDA, M.; LUDVIK, M.; TOZICKA, J. NEURONNET: Siamese Network for Anomaly Detection. *Tech. report in DCASE2020 Challenge Task 2*. 2020.

84. JMILA, Houda; IBN KHEDHER, Mohamed; BLANC, Gregory; EL YACOUBI, Mounim. Siamese Network Based Feature Learning for Improved Intrusion Detection. In: 2019, pp. 377–389. Available from DOI: 10.1007/978-3-030-36708-4_31.

85. NGUYEN, Minh-Nghia; VIEN, Ngo Anh. *Scalable and Interpretable One-class SVMs with Deep Learning and Random Fourier features*. 2018. Available from arXiv: 1804.04888 [cs.LG].

86. ANDREWS, Jerone; MORTON, Edward; GRIFFIN, Lewis. Detecting anomalous data using auto-encoders. *International Journal of Machine Learning and Computing*. 2016, vol. 6, p. 21.

87. ERFANI, Sarah M.; RAJASEGARAR, Sutharshan; KARUNASEKERA, Shanika; LECKIE, Christopher. High-Dimensional and Large-Scale Anomaly Detection Using a Linear One-Class SVM with Deep Learning. *Pattern Recogn.* 2016, vol. 58, no. C, pp. 121–134. ISSN 0031-3203. Available from DOI: 10.1016/j.patcog.2016.03.028.

88. VERCRUYSSEN, Vincent; MEERT, Wannes; DAVIS, Jesse. Transfer learning for time series anomaly detection. In: IAL ECML PKDD, 2017.

89. LIANG, Peng; YANG, Hai-Dong; CHEN, Wen-Si; XIAO, Si-Yuan; LAN, Zhao-Ze. Transfer learning for aluminium extrusion electricity consumption anomaly detection via deep neural networks. *International Journal of Computer Integrated Manufacturing*. 2017, vol. 31, pp. 1–10. Available from DOI: 10.1080/0951192X.2017.1363410.

90. ANDREWS, Jerone; TANAY, Thomas; MORTON, Edward; GRIFFIN, Lewis. Transfer Representation-Learning for Anomaly Detection. In: 2016.

91. WU, Ruobing; WANG, Baoyuan; WANG, Wenping; YU, Yizhou. *Harvesting Discriminative Meta Objects with Deep CNN Features for Scene Classification*. 2015. Available from arXiv: 1510.01440 [cs.CV].

92. SCHÖLKOPF, Bernhard; WILLIAMSON, Robert; SMOLA, Alex; SHAWE-TAYLOR, John; PLATT, John. Support Vector Method for Novelty Detection. In: 1999, vol. 12, pp. 582–588.

93. BOUNSIAR, Abdenour; MADDEN, M. Kernels for One-Class Support Vector Machines. *2014 International Conference on Information Science & Applications (ICISA)*. 2014, pp. 1–4.

94. TAX, David; DUIN, Robert. Support Vector Data Description. *Machine Learning*. 2004, vol. 54, pp. 45–66. Available from DOI: 10.1023/B:MACH.0000008084.60811.49.

95. SCHÖLKOPF, Bernhard; PLATT, John; SHAWE-TAYLOR, John; SMOLA, Alexander; WILLIAMSON, Robert. Estimating Support of a High-Dimensional Distribution. *Neural Computation*. 2001, vol. 13, pp. 1443–1471. Available from DOI: 10.1162/089976601750264965.

96. SEEBÖCK, Philipp; WALDSTEIN, Sebastian; KLIMSCHA, Sophie; GERENDAS, Bianca S.; DONNER, René; SCHLEGL, Thomas; SCHMIDT-ERFURTH, Ursula; LANGS, Georg. *Identifying and Categorizing Anomalies in Retinal Imaging Data*. 2016. Available from arXiv: 1612.00686 [cs.LG].

97. DOSHI, Keval; YILMAZ, Yasin. *Continual Learning for Anomaly Detection in Surveillance Videos*. 2020. Available from arXiv: 2004.07941 [cs.CV].

98. GERYCH, W.; AGU, E.; RUNDENSTEINER, E. Classifying Depression in Imbalanced Datasets Using an Autoencoder- Based Anomaly Detection Approach. In: *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*. Los Alamitos, CA, USA: IEEE Computer Society, 2019, pp. 124–127. ISSN 2325-6516. Available from DOI: 10.1109/ICOSC.2019.8665535.

99. MABU, Shingo; FUJITA, Kohki; KUREMOTO, Takashi. Disaster Area Detection from Synthetic Aperture Radar Images Using Convolutional Autoencoder and One-class SVM. *Journal of Robotics, Networking and Artificial Life*. 2019, vol. 6, p. 48. Available from DOI: 10.2991/jrnal.k.190601.001.

100. SOHAIB, Muhammad; KIM, Cheol-Hong; KIM, Jongmyon. A Hybrid Feature Model and Deep-Learning-Based Bearing Fault Diagnosis. *Sensors*. 2017, vol. 17, p. 2876. Available from DOI: 10.3390/s17122876.

101. TANG, T; MHAMDI, Lotfi; ZAIDI, Syed Ali Raza; EL-MOUSA, Fadi; MCLERNON, D; GHOGHO, Mounir. A Deep Learning Approach Combining Auto-encoder with One-class SVM for DDoS Attack Detection in SDNs. In: 2019.

102. ELSAYED, Mahmoud; LE-KHAC, Nhien-An; DEV, Soumyabrata; JURCUT, Anca. Network Anomaly Detection Using LSTM Based Autoencoder. In: 2020.

103. KANG, Xu; SONG, Bin; SUN, Fengyao. A Deep Similarity Metric Method Based on Incomplete Data for Traffic Anomaly Detection in IoT. *Applied Sciences*. 2019, vol. 9, p. 135. Available from DOI: 10.3390/app9010135.

104. FOSTER, David. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose and Play*. O'Reilly Media, Inc, USA, 2019.

105. KARRAS, Tero; LAINE, Samuli; AILA, Timo. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2019. Available from arXiv: 1812.04948 [cs.NE].

106. BROWN, Tom B.; MANN, Benjamin; RYDER, Nick; SUBBIAH, Melanie; KAPLAN, Jared; DHARIWAL, Prafulla; NEELAKANTAN, Arvind; SHYAM, Pranav; SASTRY, Girish; ASKELL, Amanda; AGARWAL, Sandhini; HERBERT-VOSS, Ariel; KRUEGER, Gretchen; HENIGHAN, Tom; CHILD, Rewon; RAMESH, Aditya; ZIEGLER, Daniel M.; WU, Jeffrey; WINTER, Clemens; HESSE, Christopher; CHEN, Mark; SIGLER, Eric; LITWIN, Mateusz; GRAY, Scott; CHESS, Benjamin; CLARK, Jack; BERNER, Christopher; MCCANDLISH, Sam; RADFORD, Alec; SUTSKEVER, Ilya; AMODEI, Dario. *Language Models are Few-Shot Learners*. 2020. Available from arXiv: 2005.14165 [cs.CL].

107. DONAHUE, Jeff; KRÄHENBÜHL, Philipp; DARRELL, Trevor. *Adversarial Feature Learning*. 2017. Available from arXiv: 1605.09782 [cs.LG].

108. LUPO, Francesco. *Variational Autoencoder for unsupervised anomaly detection*. 2019. MA thesis. POLITECNICO DI TORINO.

109. VOYNOV, Andrey; BABENKO, Artem. *Unsupervised Discovery of Interpretable Directions in the GAN Latent Space*. 2020. Available from arXiv: 2002.03754 [cs.LG].

110. ŠKVÁRA, Vít; PEVNÝ, Tomáš; ŠMÍDL, Václav. *Are generative deep models for novelty detection truly better?* 2018. Available from arXiv: 1807.05027 [cs.LG].

111. KINGMA, Diederik P; WELLING, Max. *Auto-Encoding Variational Bayes*. 2014. Available from arXiv: 1312.6114 [stat.ML].

112. GOODFELLOW, Ian J.; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David; OZAIR, Sherjil; COURVILLE, Aaron; BENGIO, Yoshua. *Generative Adversarial Networks*. 2014. Available from arXiv: 1406.2661 [stat.ML].

113. SCHLEGL, Thomas; SEEBÖCK, Philipp; WALDSTEIN, Sebastian M.; SCHMIDT-ERFURTH, Ursula; LANGS, Georg. *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. 2017. Available from arXiv: 1703.05921 [cs.CV].

114. TAMIR, Mike [online]. 2018 [visited on 2020-12-13]. Available from: https://twitter.com/miketamir/status/1069055811115237376.

115. ARJOVSKY, Martin; CHINTALA, Soumith; BOTTOU, Léon. *Wasserstein GAN*. 2017. Available from arXiv: 1701.07875 [stat.ML].

116. MATTIA, Federico Di; GALEONE, Paolo; SIMONI, Michele De; GHELFI, Emanuele. *A Survey on GANs for Anomaly Detection*. 2019. Available from arXiv: 1906.11632 [cs.LG].

117. DUMOULIN, Vincent; BELGHAZI, Ishmael; POOLE, Ben; MASTROPIETRO, Olivier; LAMB, Alex; ARJOVSKY, Martin; COURVILLE, Aaron. *Adversarially Learned Inference*. 2017. Available from arXiv: 1606.00704 [stat.ML].

118. ZENATI, Houssam; FOO, Chuan Sheng; LECOUAT, Bruno; MANEK, Gaurav; CHANDRASEKHAR, Vijay Ramaseshan. *Efficient GAN-Based Anomaly Detection*. 2019. Available from arXiv: 1802.06222 [cs.LG].

119. VIANA, Jaime; ROSA, Ezequiel de la; VANDE VYVERE, Thijs; ROBBEN, David; SIMA, Diana. *Unsupervised 3D Brain Anomaly Detection*. 2020.

120. BENSON, Sean; BEETS-TAN, Regina. GAN-based anomaly detection in multi-modal MRI images. *bioRxiv*. 2020. Available from DOI: 10.1101/2020.07.10.197087.

121. LU, Yuchen; XU, Peng. *Anomaly Detection for Skin Disease Images Using Variational Autoencoder*. 2018. Available from arXiv: 1807.01349 [cs.LG].

122. MISIK, Adam. *Potential Analysis of Wasserstein GAN as an Anomaly Detection Method for Industrial Images*. 2020.

123. LIANG, Haoran; SONG, Lei; WANG, Jianxing; GUO, Lili; LI, Xuzhi; LIANG, Ji. Robust unsupervised anomaly detection via multi-time scale DCGANs with forgetting mechanism for industrial multivariate time series. *Neurocomputing*. 2020, vol. 490. Available from DOI: 10.1016/j.neucom.2020.10.084.

124. LYU, Yang; HAN, Zhiwei; ZHONG, Junping; LI, Changjiang; LIU, Zhigang. A Generic Anomaly Detection of Catenary Support Components Based on Generative Adversarial Networks. *IEEE Transactions on Instrumentation and Measurement*. 2020.

125. KATAFUCHI, Ryoya; TOKUNAGA, Terumasa. *Image-based plant disease diagonasis with unsupervised anomaly detection based on reconstructability of colors*. 2020.

126. MASAKI, Ayaka; NAGUMO, Kent; LAMSAL, Bikash; OIWA, Kosuke; NOZAWA, Akio. Anomaly detection in facial skin temperature using variational autoencoder. *Artificial Life and Robotics*. 2020, pp. 1–7. Available from DOI: 10.1007/s10015-020-00634-2.

127. DUMAGPI, Joanna; JUNG, Woo-Young; JEONG, Yong-Jin. A New GAN-Based Anomaly Detection (GBAD) Approach for Multi-Threat Object Classification on Large-Scale X-Ray Security Images. *IEICE Transactions on Information and Systems*. 2020, vol. E103.D, pp. 454–458. Available from DOI: 10.1587/transinf.2019EDL8154.

128. XU, Haowen; FENG, Yang; CHEN, Jie; WANG, Zhaogang; QIAO, Honglin; CHEN, Wenxiao; ZHAO, Nengwen; LI, Zeyan; BU, Jiahao; LI, Zhihan; AL., et. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. 2018. ISBN 9781450356398. Available from DOI: 10.1145/3178876.3185996.

129. LOPEZ-MARTIN, Manuel; CARRO, Belén; SANCHEZ-ESGUEVILLAS, Antonio; LLORET, Jaime. Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT. *Sensors*. 2017, vol. 17, p. 1967. Available from DOI: 10.3390/s17091967.

130. AYGUN, R.; YAVUZ, A. Network Anomaly Detection with Stochastically Improved Autoencoder Based Models. In: 2017, pp. 193–198. Available from DOI: 10.1109/CSCloud.2017.39.

131. FIORE, Ugo; SANTIS, Alfredo; PERLA, Francesca; ZANETTI, Paolo; PALMIERI, Francesco. Using Generative Adversarial Networks for Improving Classification Effectiveness in Credit Card Fraud Detection. *Information Sciences*. 2017, vol. 479. Available from DOI: 10.1016/j.ins.2017.12.030.

132. CHALAPATHY, Raghavendra; MENON, Aditya; CHAWLA, Sanjay. Anomaly Detection using One-Class Neural Networks. 2018.

133. DENG, Xiaogang; ZHANG, Zheng. Nonlinear Chemical Process Fault Diagnosis Using Ensemble Deep Support Vector Data Description. *Sensors*. 2020, vol. 20, p. 4599. Available from DOI: 10.3390/s20164599.

134. HALAMÍČEK, Martin. *Granulometrie biomedicínských obrazů* [*online*]. 2007 [cit. 2020-12-28]. Available also from: `Dostupn%C3%A9%20z%20WWW%20%3Chttps://is.muni.cz/th/ai27b/%3E`. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Supervised by Pavel MATULA.

A Electronic appendix overview

The electronic appendix consists of 3 archives – **Source**, **Saved_Models** and **Data**. The **Source** archive contains several folders:

- *Data* and *DataBigan*. The folders where the data sets reside. These folders are initially empty, as the contents of the **Data** archive belong in here. The data sets are in a separate archive due to their sheer size.
- *Model_Saves*. Also initially empty, this is where the **Saved_Models** archive should be unpacked, containing the trained weights and architectures of the best performing models. As these are massive in size, they also come in a separate archive.
- *Models*. The source classes of the tested models reside here. Also contains the specific loss functions used.
- *Reconstructed*. A number of reconstructed or generated images can be found in this folder, for each of the approaches that utilize them. These reconstructions provide additional insight into the quality of the tested models.
- *Results*. This folder contains all of the plethora of graphs plotted for each approach, showcasing the achieved results and evolution of training loss and accuracy over the epochs.
- *Scripts*. These scripts were used to run the experiment driving scripts on the Metacentrum grid.
- *Thesis*. This folder contains everything needed to create this very document.

In addition, the main folder is where the experiments-related scripts themselves are. Along with a few common utility scripts, each approach has its own:

- Data preparation script, denoted by *data_prep* in its name. Their objective is to turn the provided raw images into reshaped and normalized `numpy` arrays, suitable for neural network training

and testing. Note that these scripts aren't usable, as the raw images are not a part of the appendix. Instead, the **Data** archive provides the aforementioned `numpy` arrays.

- Experiment driving script, with *tester* in its name. These scripts are the main work horses, as they take care of the whole training and testing ordeal and produce all of the performance-describing graphs. The concrete model to be used and some hyperparameter choices are governed by running these scripts with dedicated parameters.
- Defect detector-simulating script, with the *predict_tester* suffix. In their current form, they load the best trained models and perform anomaly detection on each applicable sample, while showing the wrongly classified images to the user. These came handy when figuring out problematic images and types of defects. With slight modifications, they can be transformed into fully usable automated defect detectors.

All of the scripts include thorough docstrings, explaining their usage.

B High-resolution images of apertures

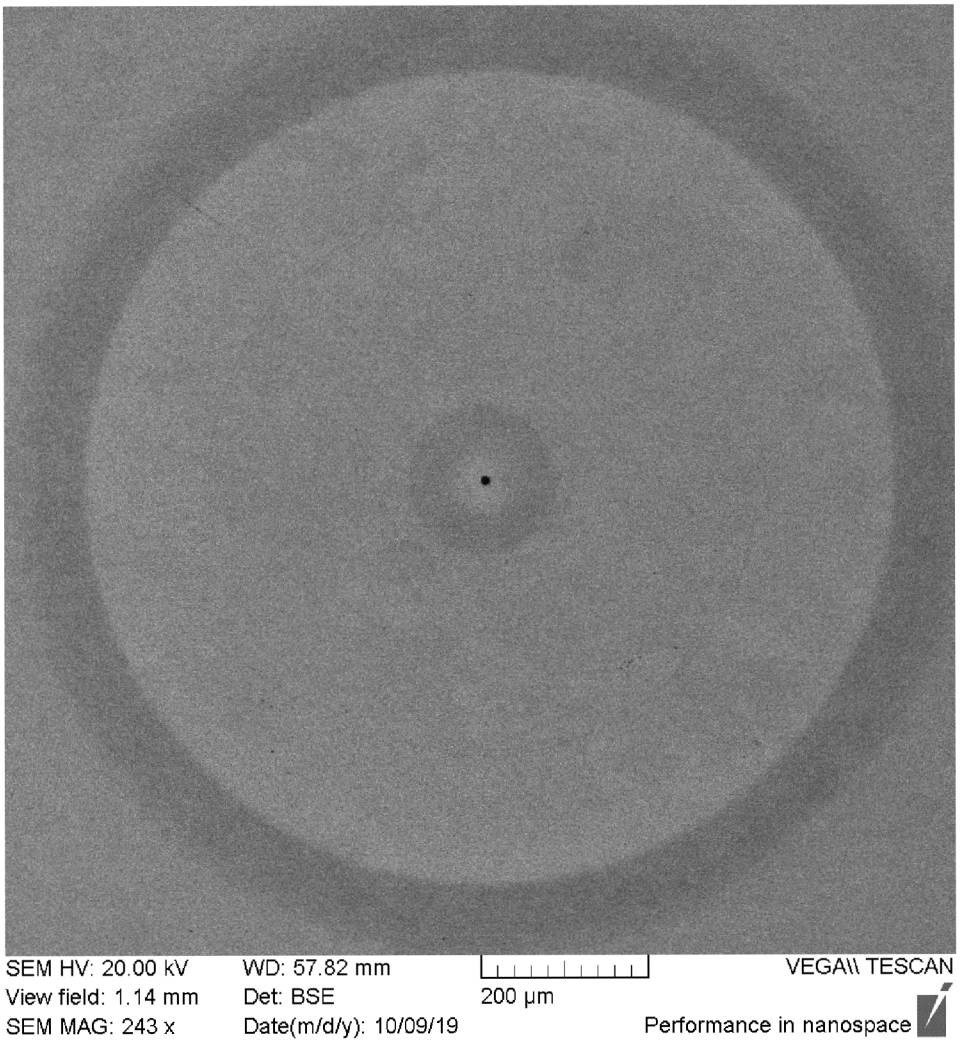


Figure B.1: Hi-res version of the example non-defective BSE image used in Figure 3.1.

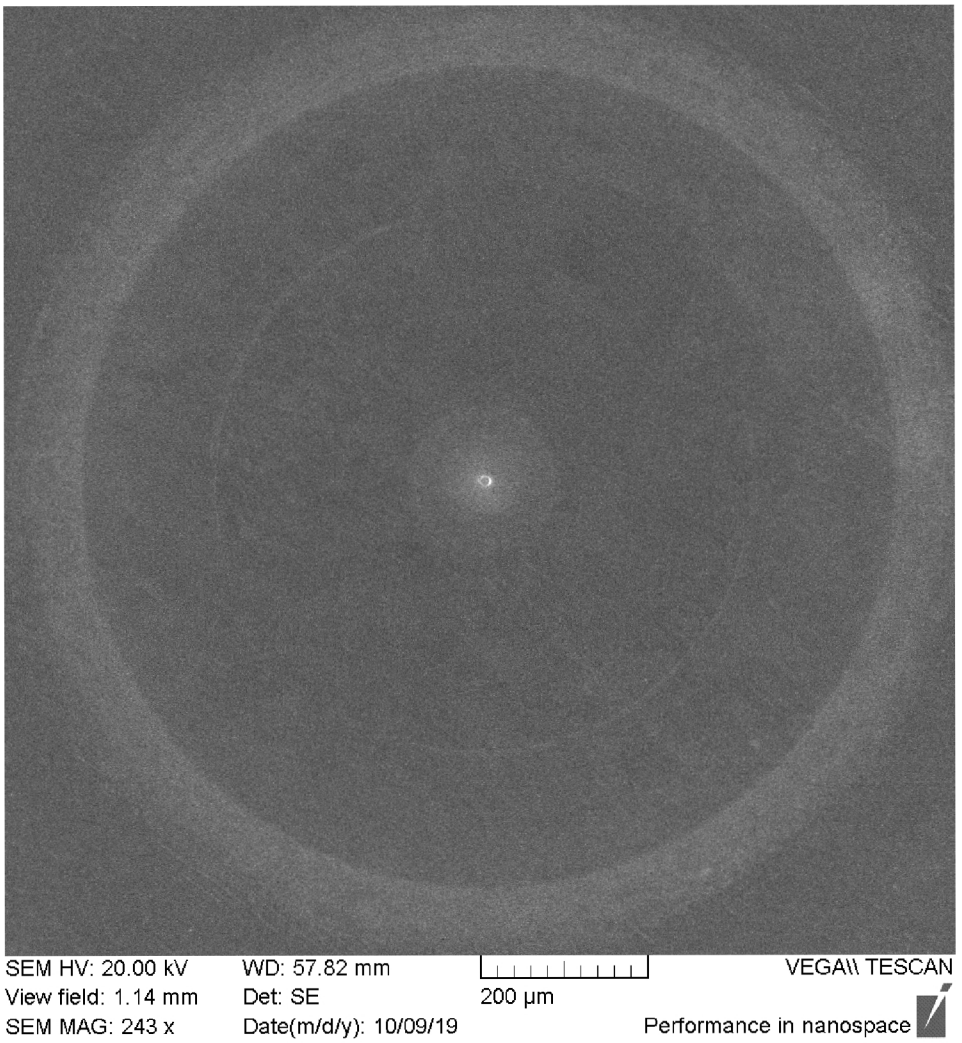


Figure B.2: Hi-res versions of the example non-defective SE image used in Figure 3.1.

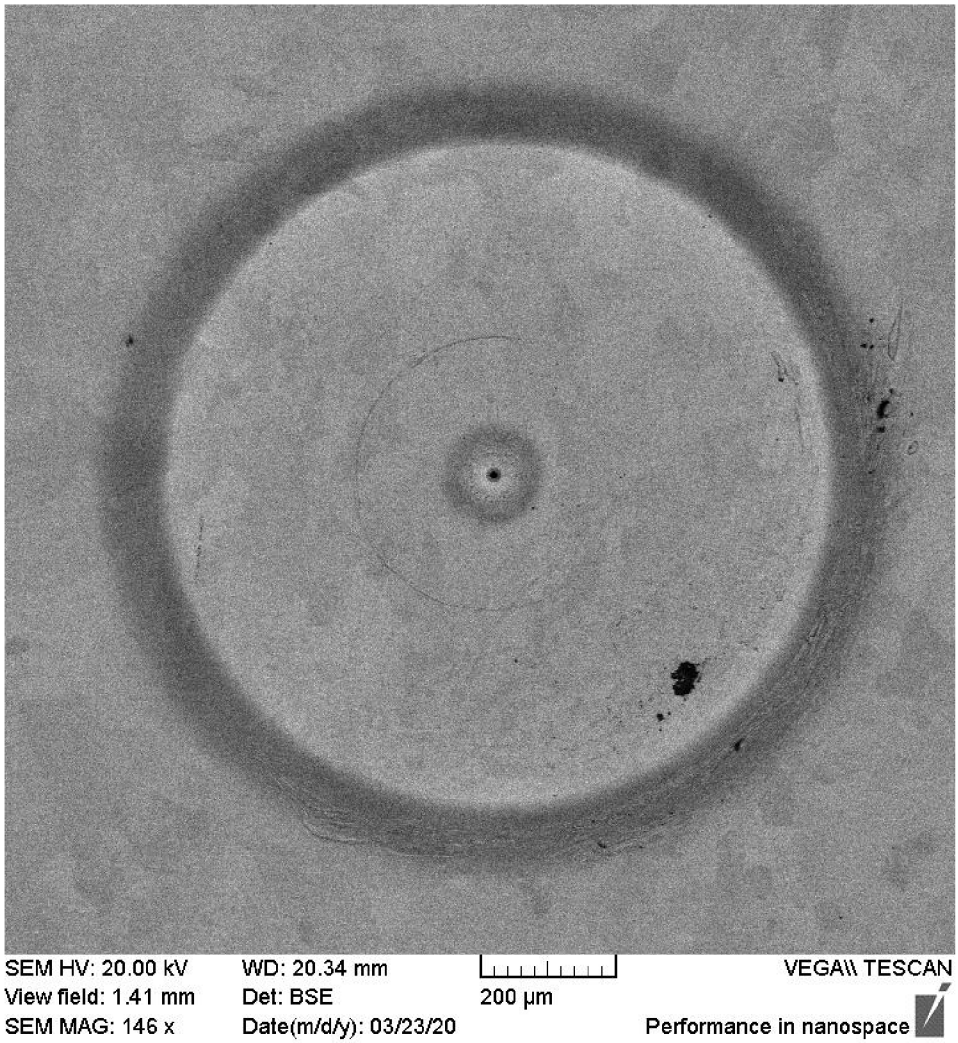


Figure B.3: Hi-res version of surface damage from Figure 3.2.

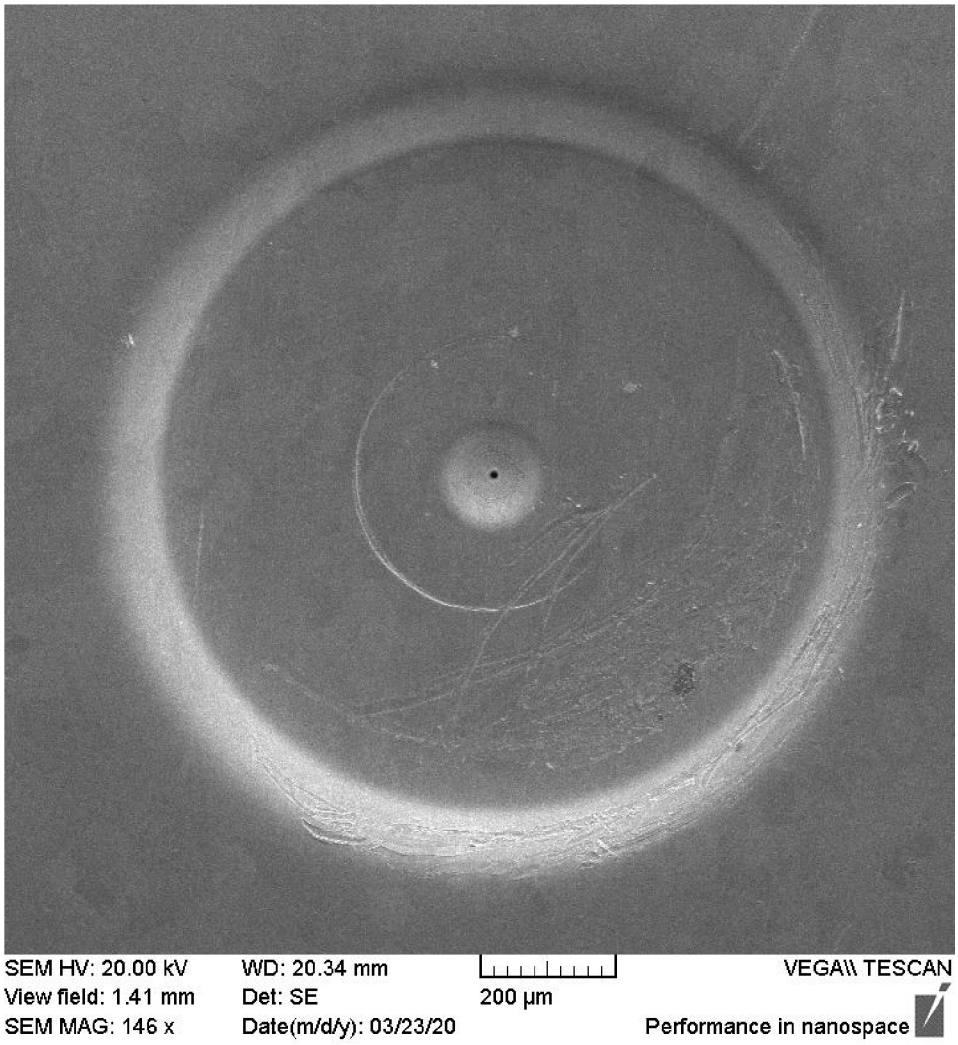


Figure B.4: Hi-res version of surface damage from Figure 3.2.

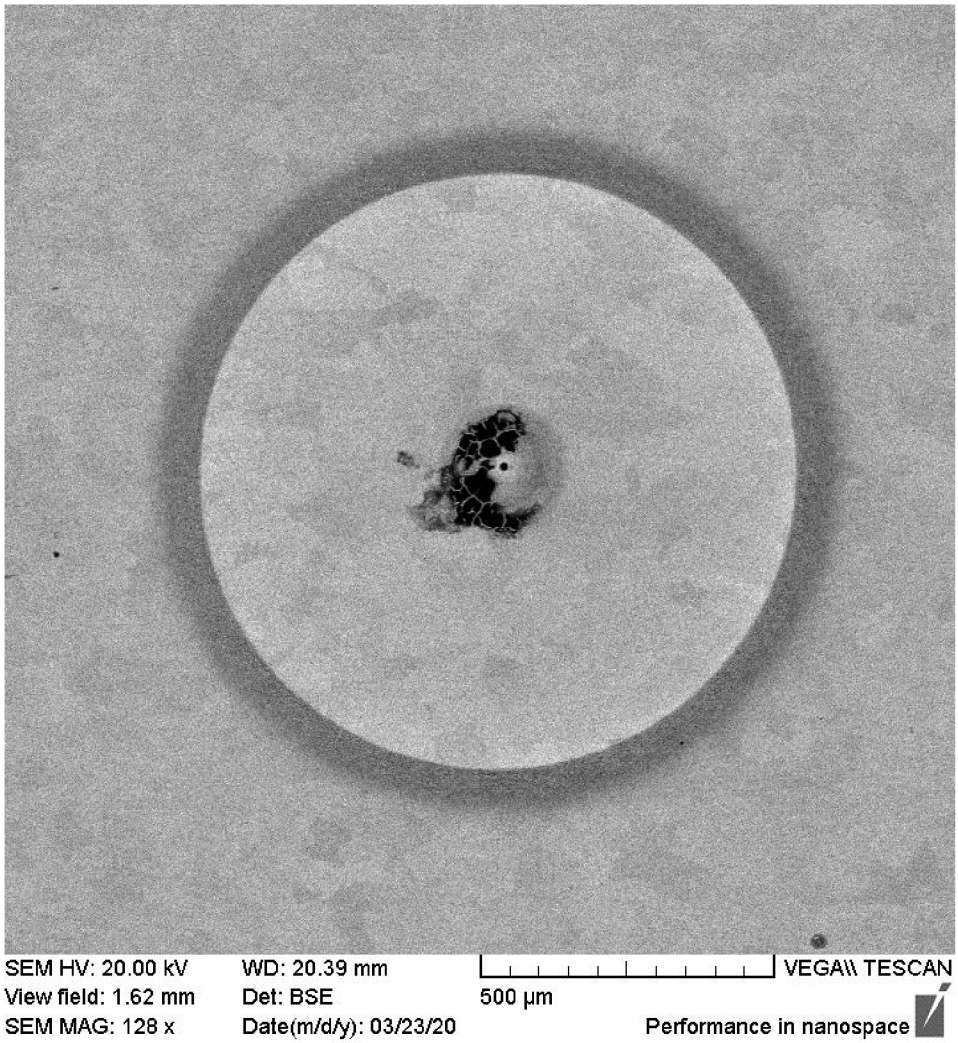


Figure B.5: Hi-res version of surface damage from Figure 3.2.

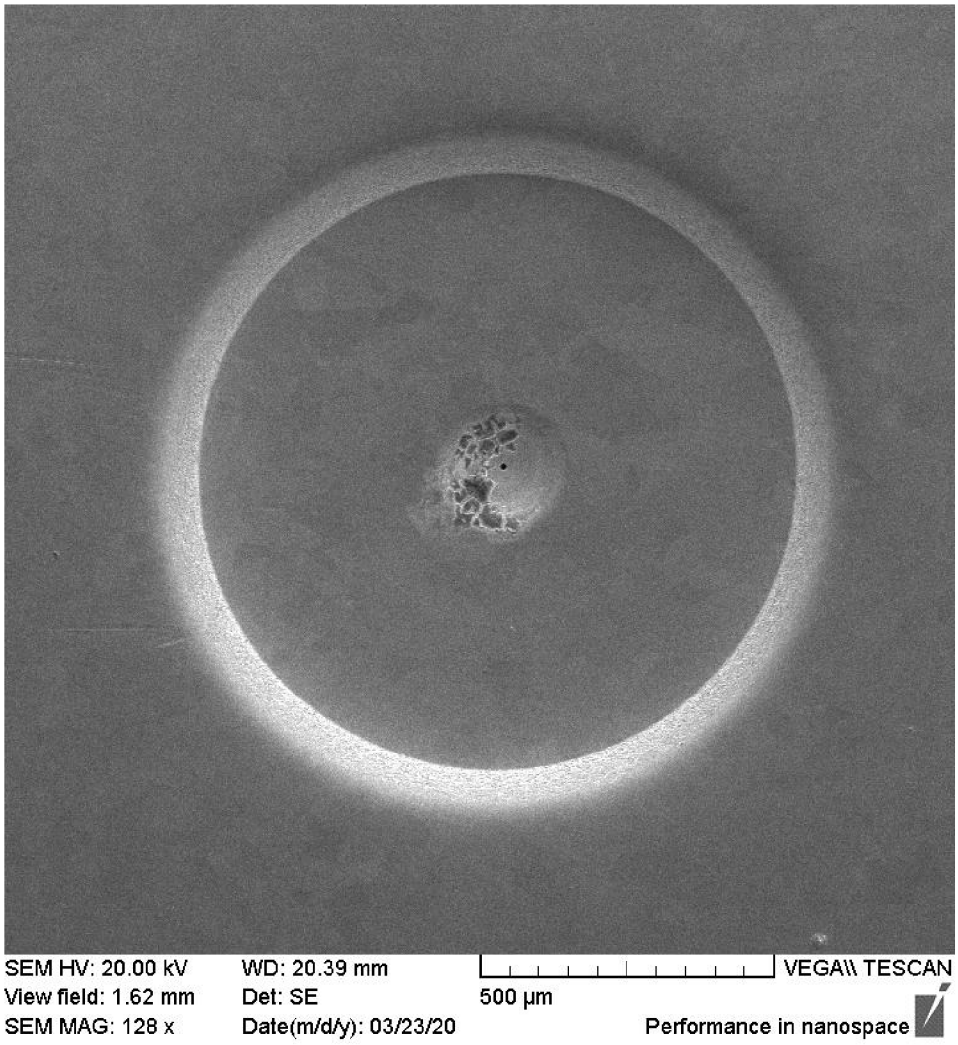


Figure B.6: Hi-res version of surface damage from Figure 3.2.

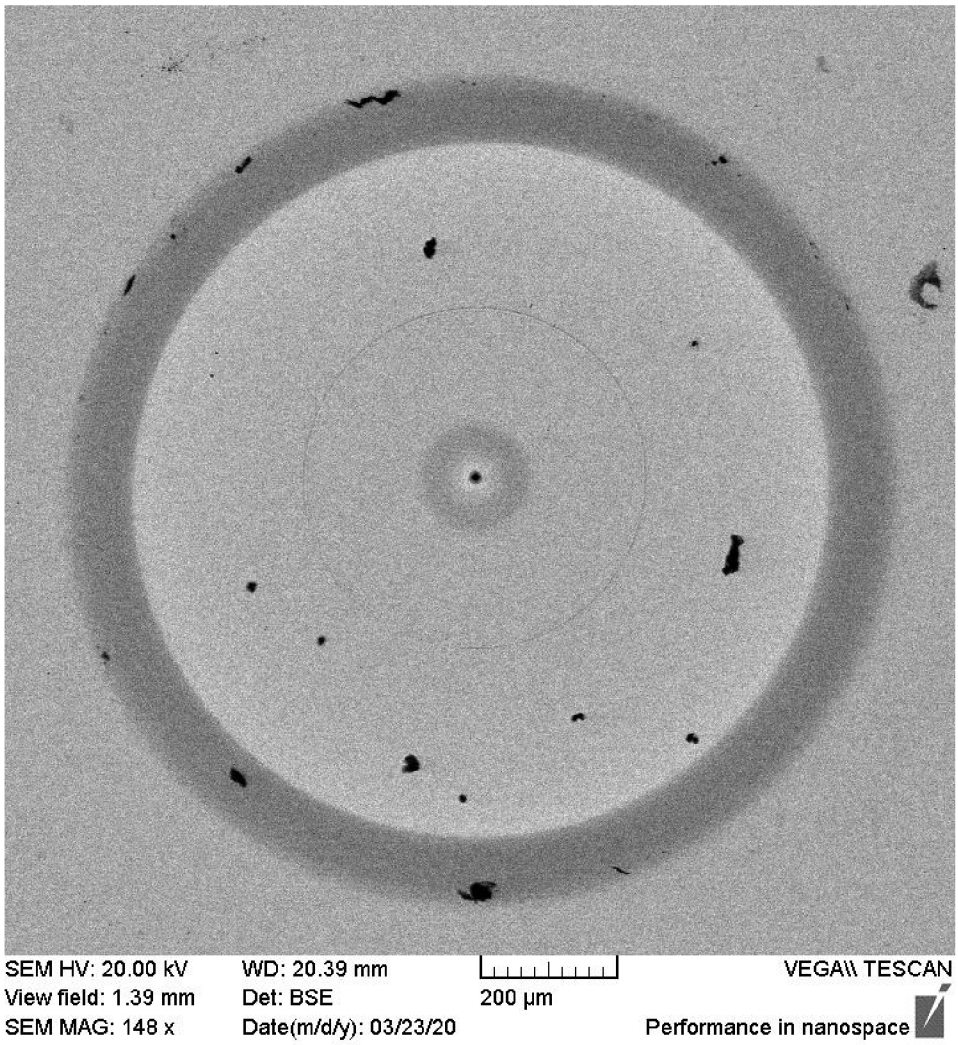


Figure B.7: Hi-res version of an aperture with substantial stains from Figure 3.3.

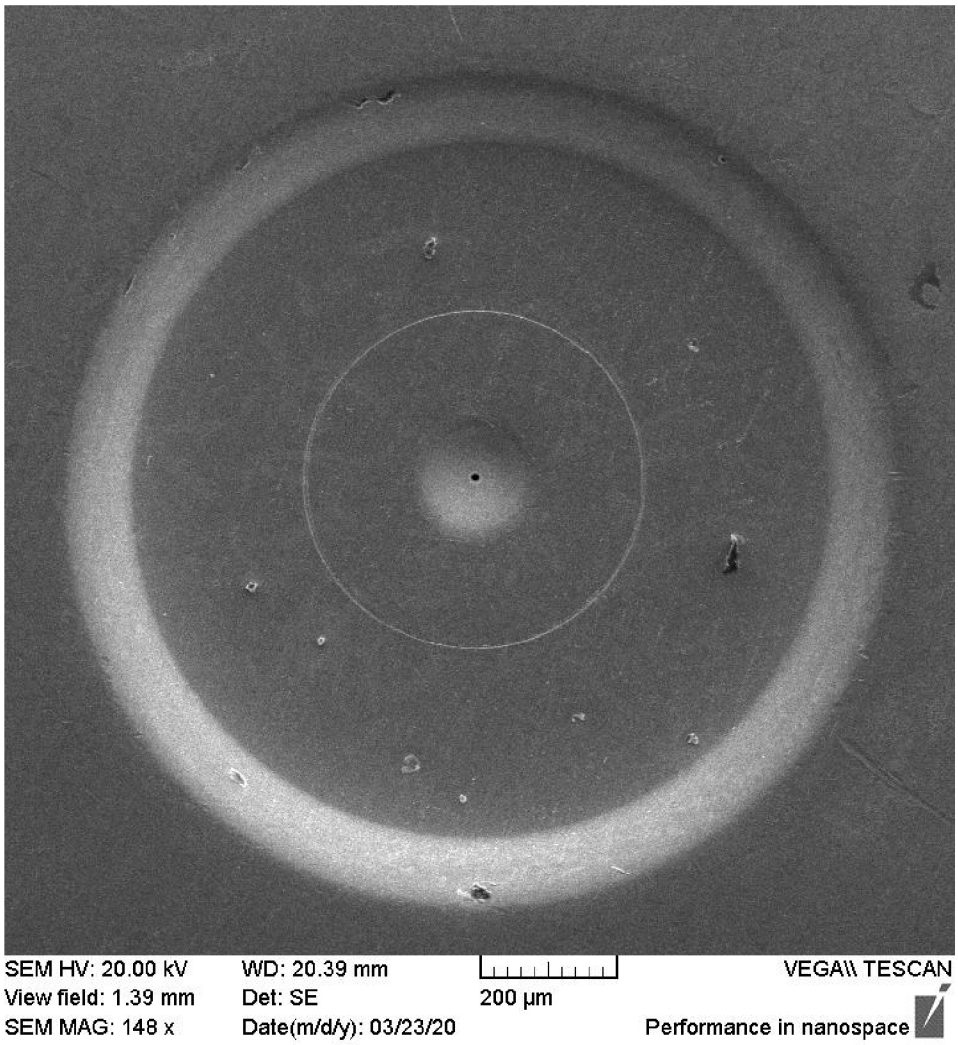


Figure B.8: Hi-res version of an aperture with substantial stains from Figure 3.3.

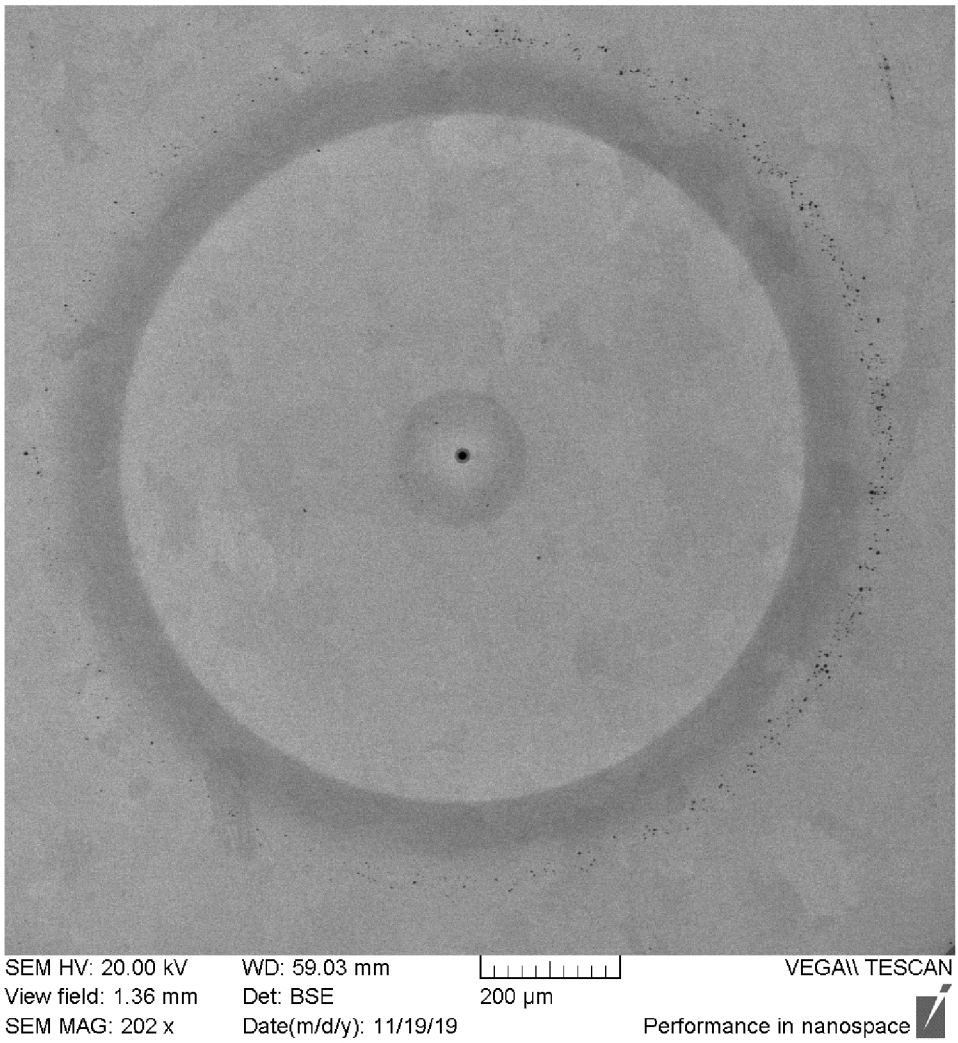


Figure B.9: Hi-res version of an aperture with commonly found tolerable stains from Figure 3.4.

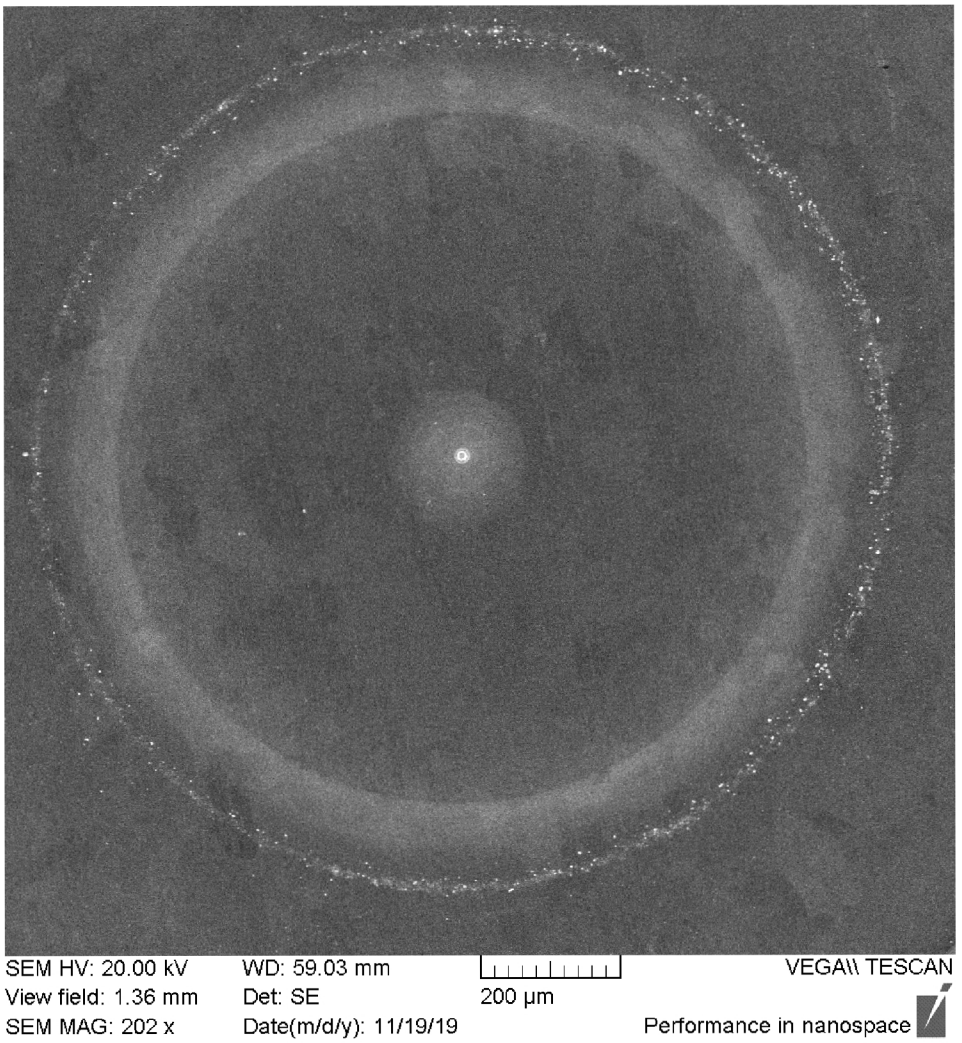


Figure B.10: Hi-res version of an aperture with commonly found tolerable stains from Figure 3.4.

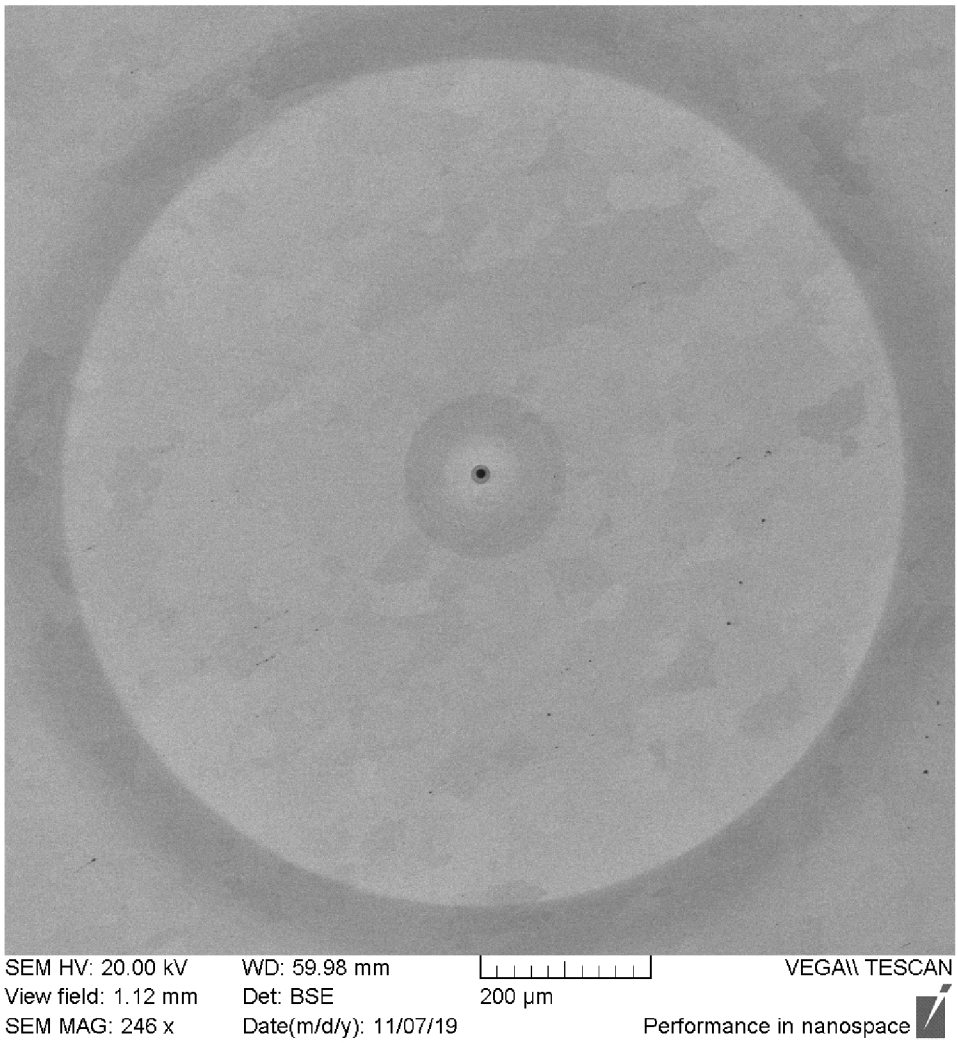


Figure B.11: Hi-res version of an aperture with commonly found tolerable stains from Figure 3.4.

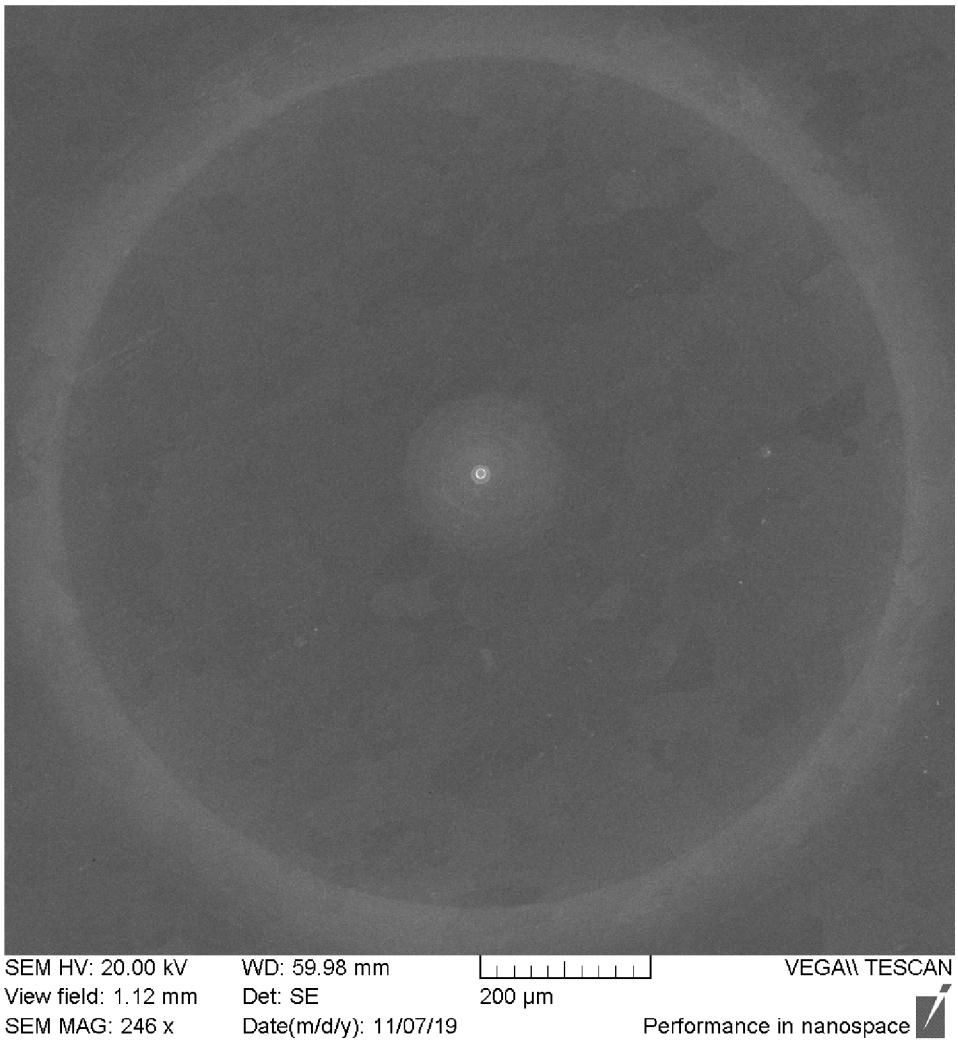


Figure B.12: Hi-res version of an aperture with commonly found tolerable stains from Figure 3.4.

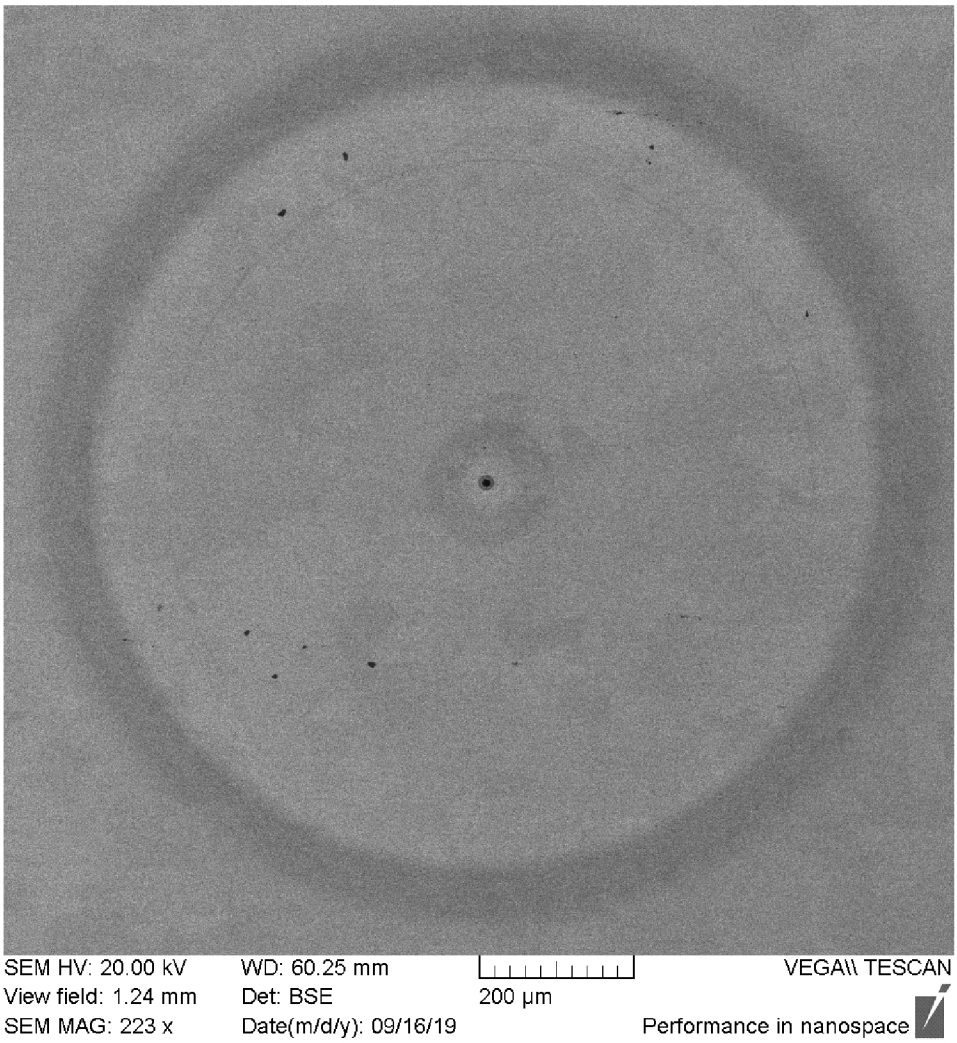


Figure B.13: Hi-res version of a tolerable aperture with slight stains from Figure 3.5.

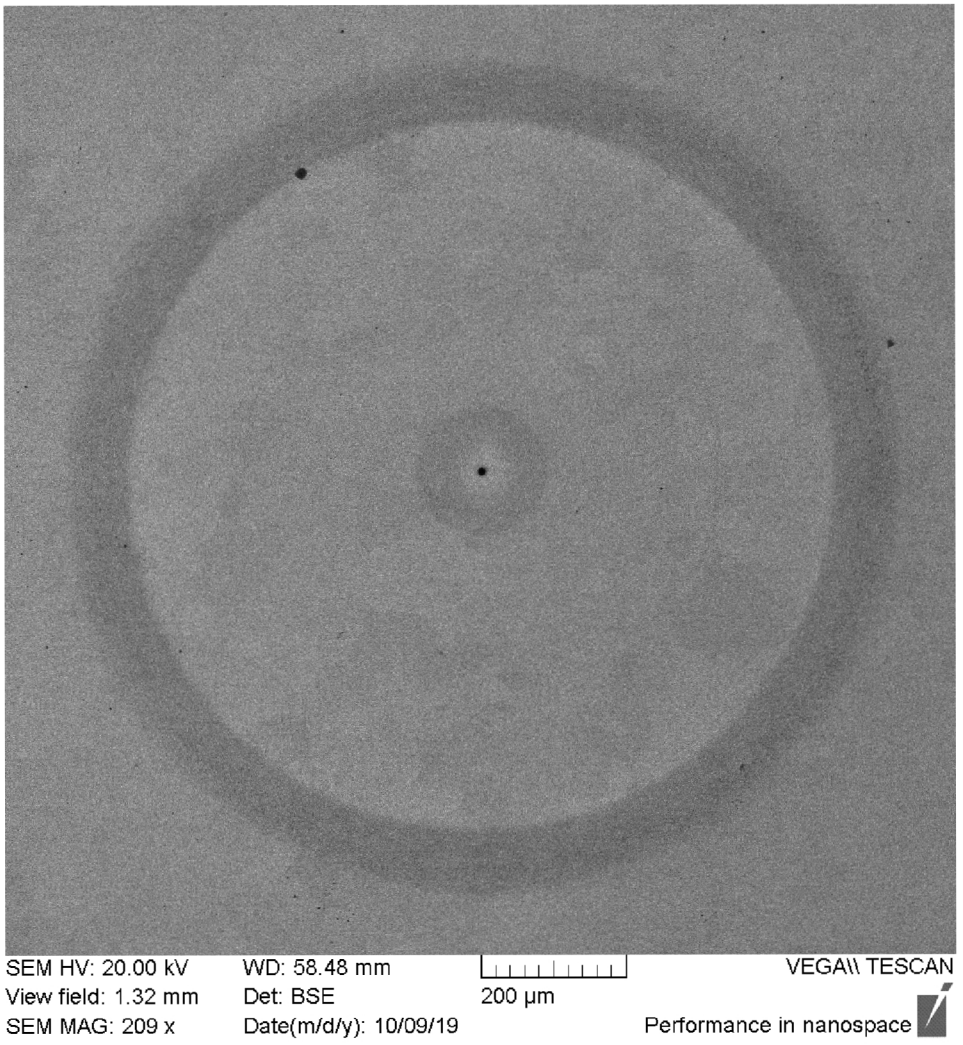


Figure B.14: Hi-res version of a tolerable aperture with slight stains from Figure 3.5.

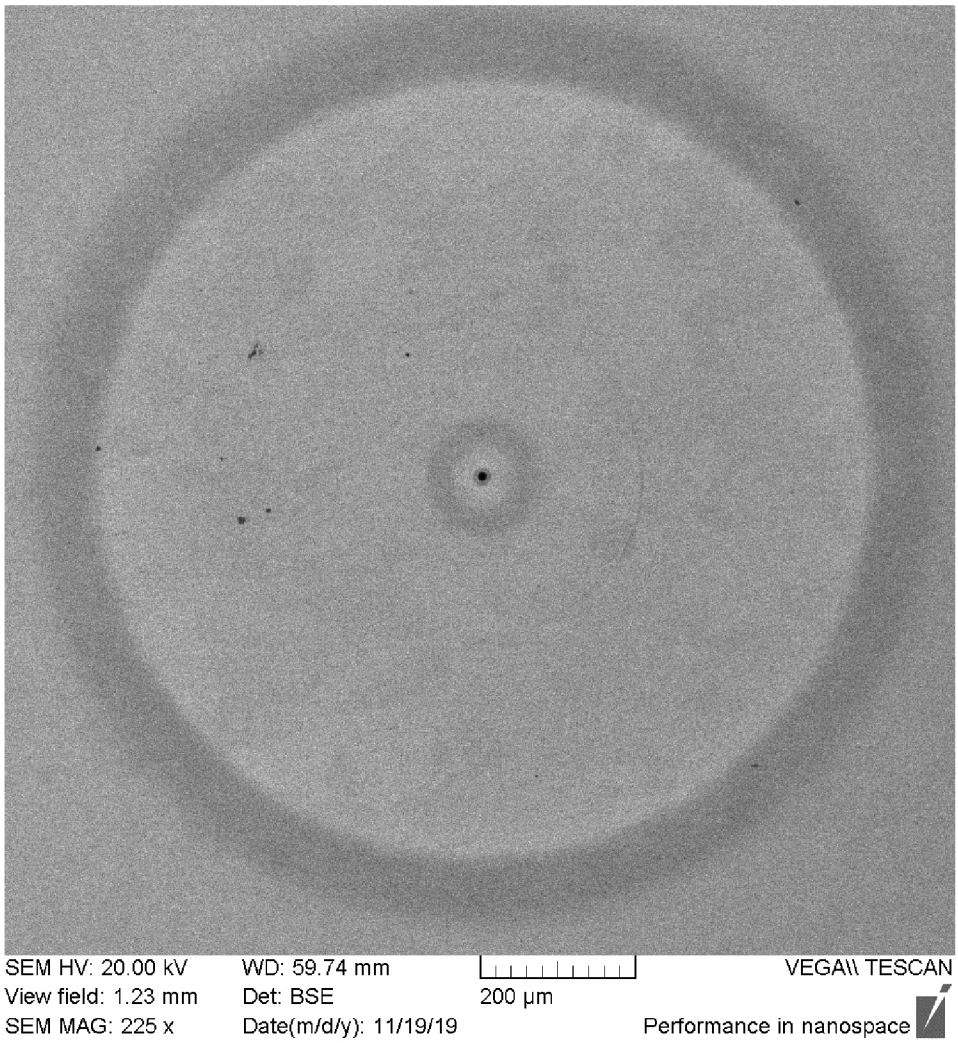


Figure B.15: Hi-res version of a tolerable aperture with slight stains from Figure 3.5.

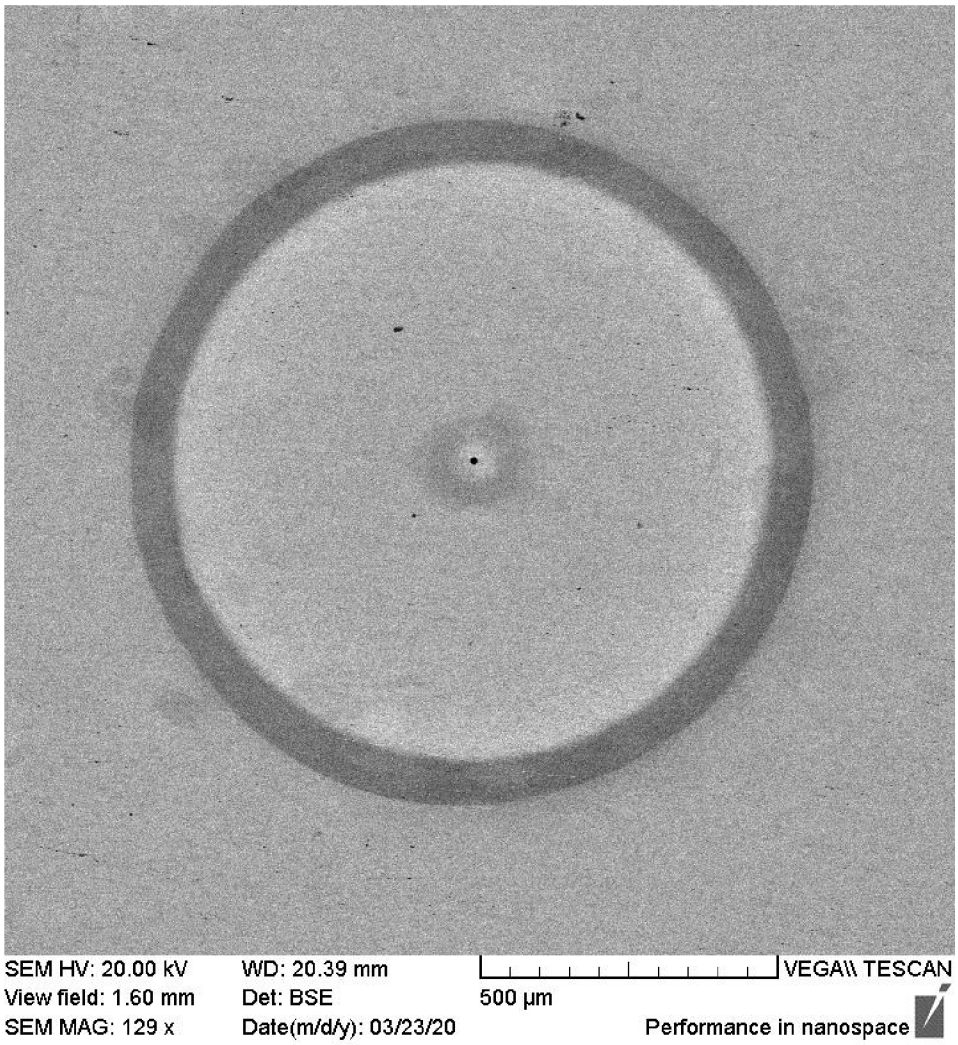


Figure B.16: Hi-res version of a defective aperture with slight stains from Figure 3.5.

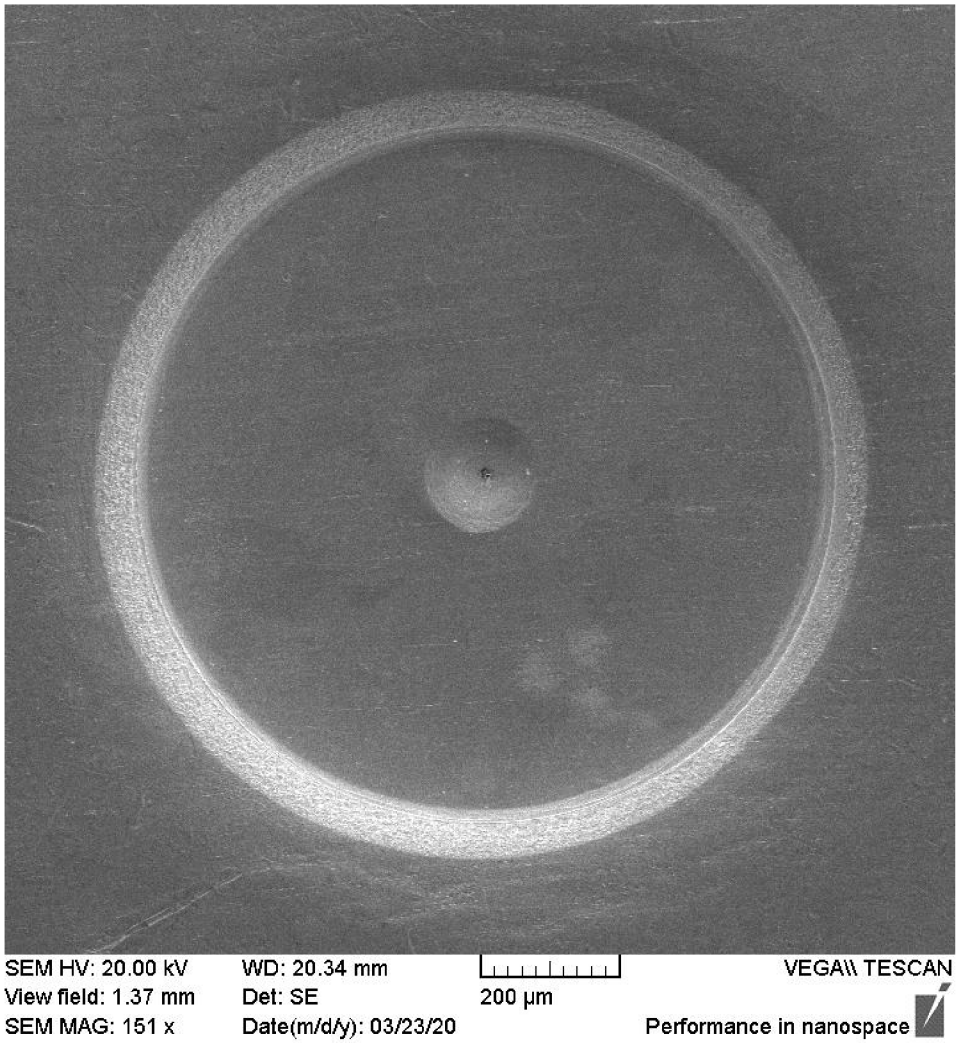


Figure B.17: Hi-res version of a clogged aperture from Figure 3.6.

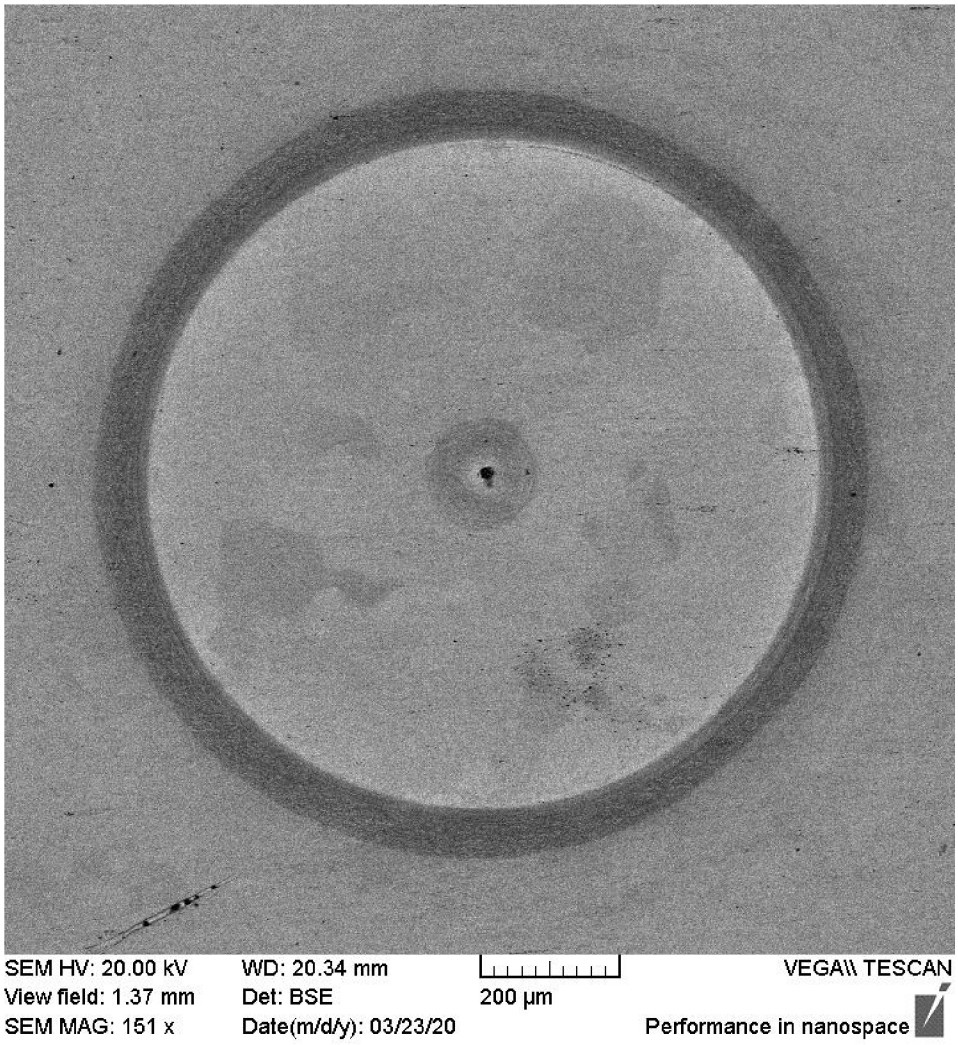


Figure B.18: Hi-res version of a clogged aperture.

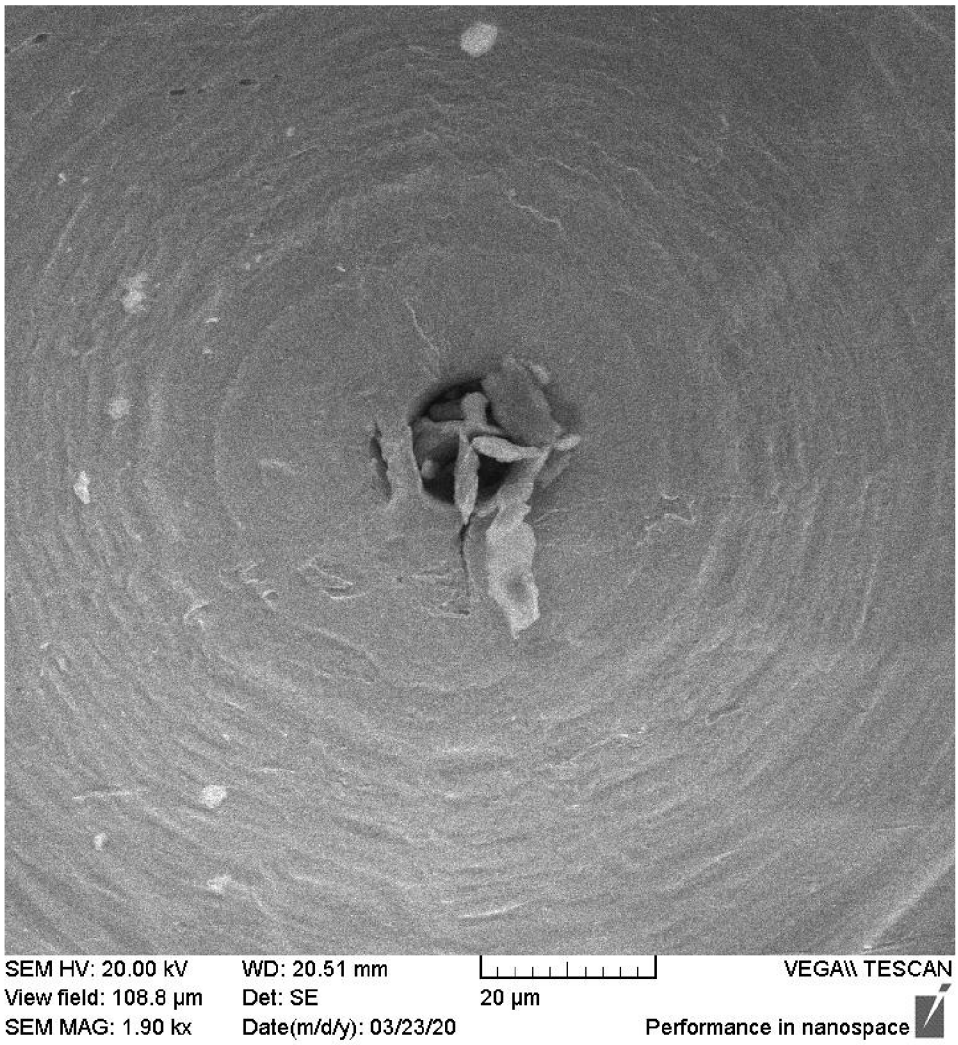


Figure B.19: Hi-res version of a magnified image of a clogged aperture from Figure 3.6.

C Prototype apertures

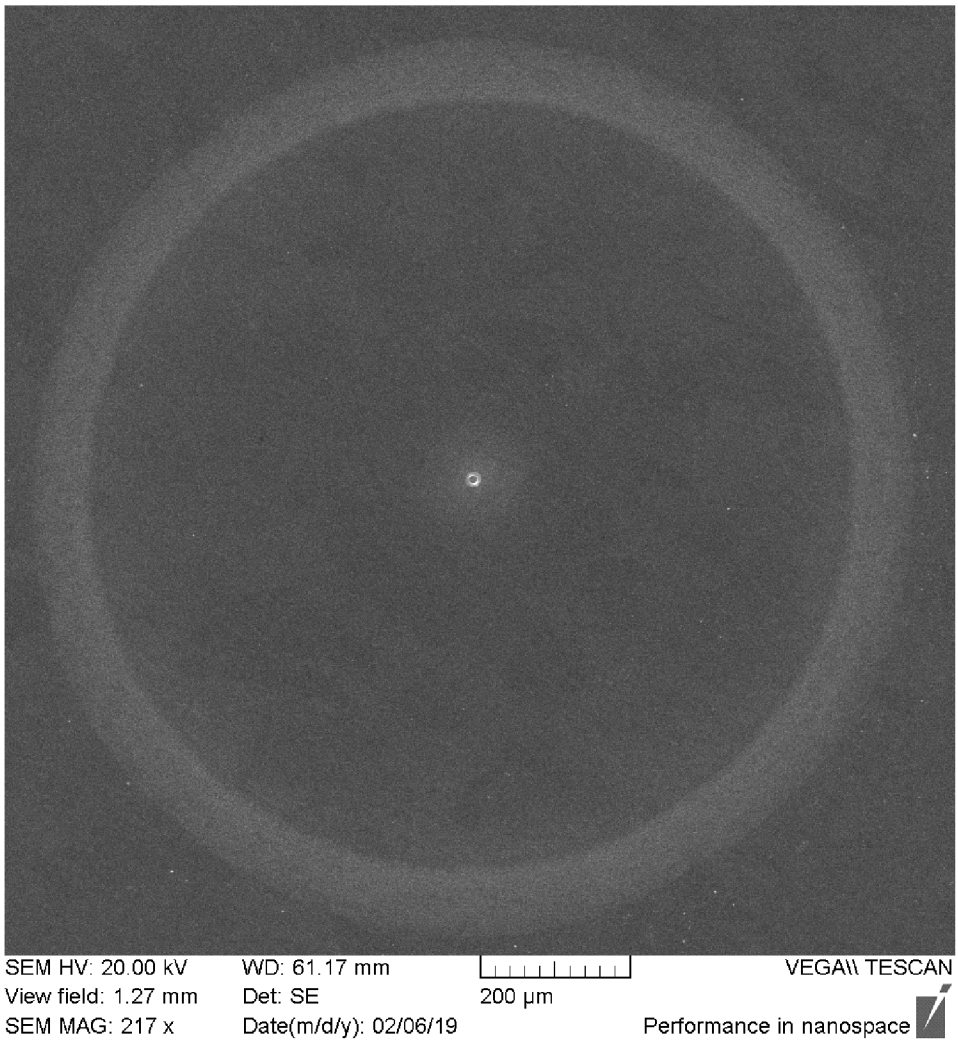


Figure C.1: Prototype image 1

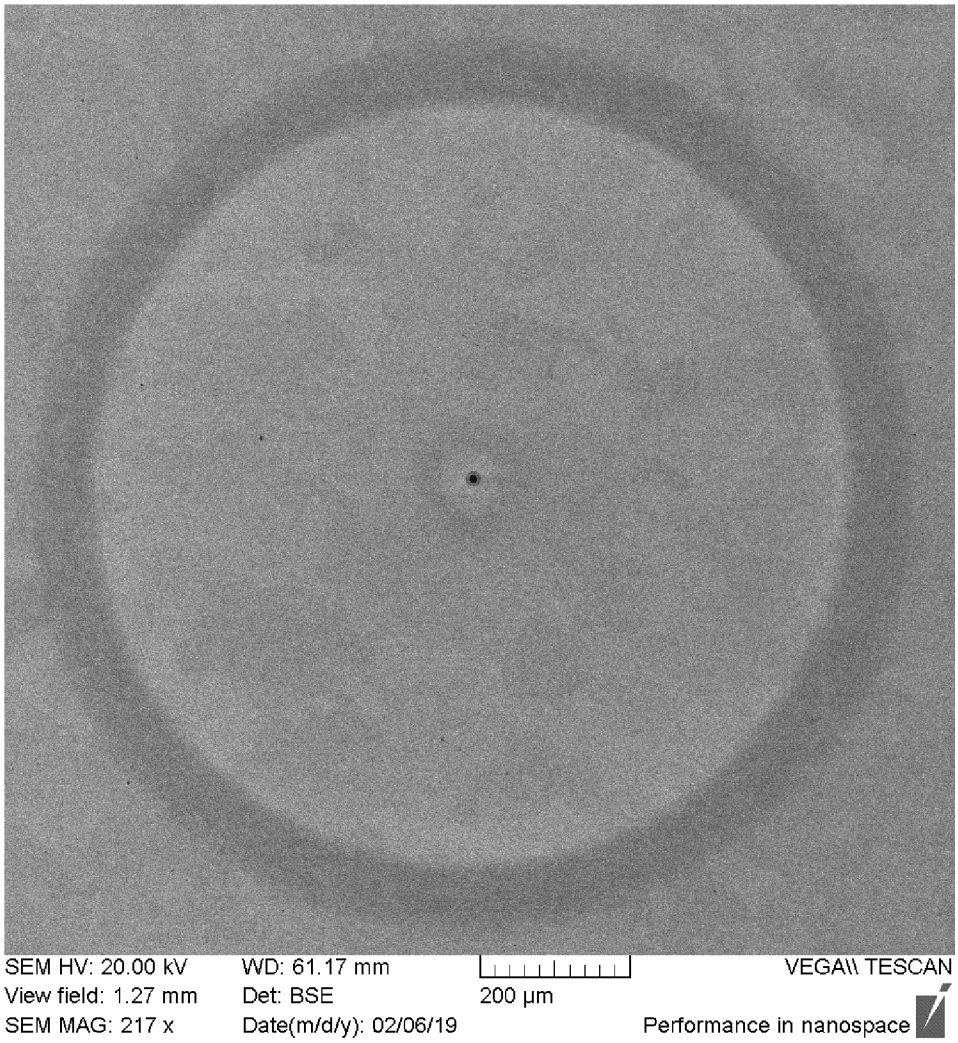


Figure C.2: Prototype image 1

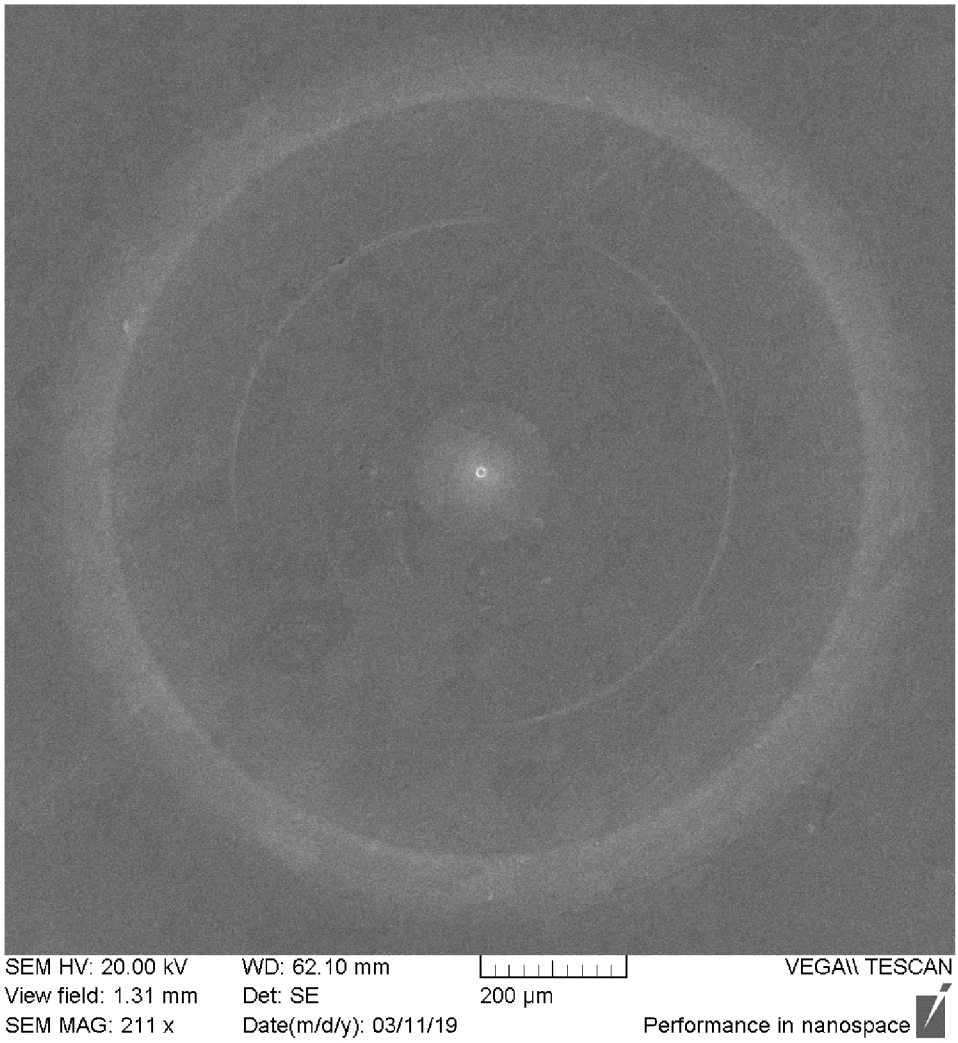


Figure C.3: Prototype image 2

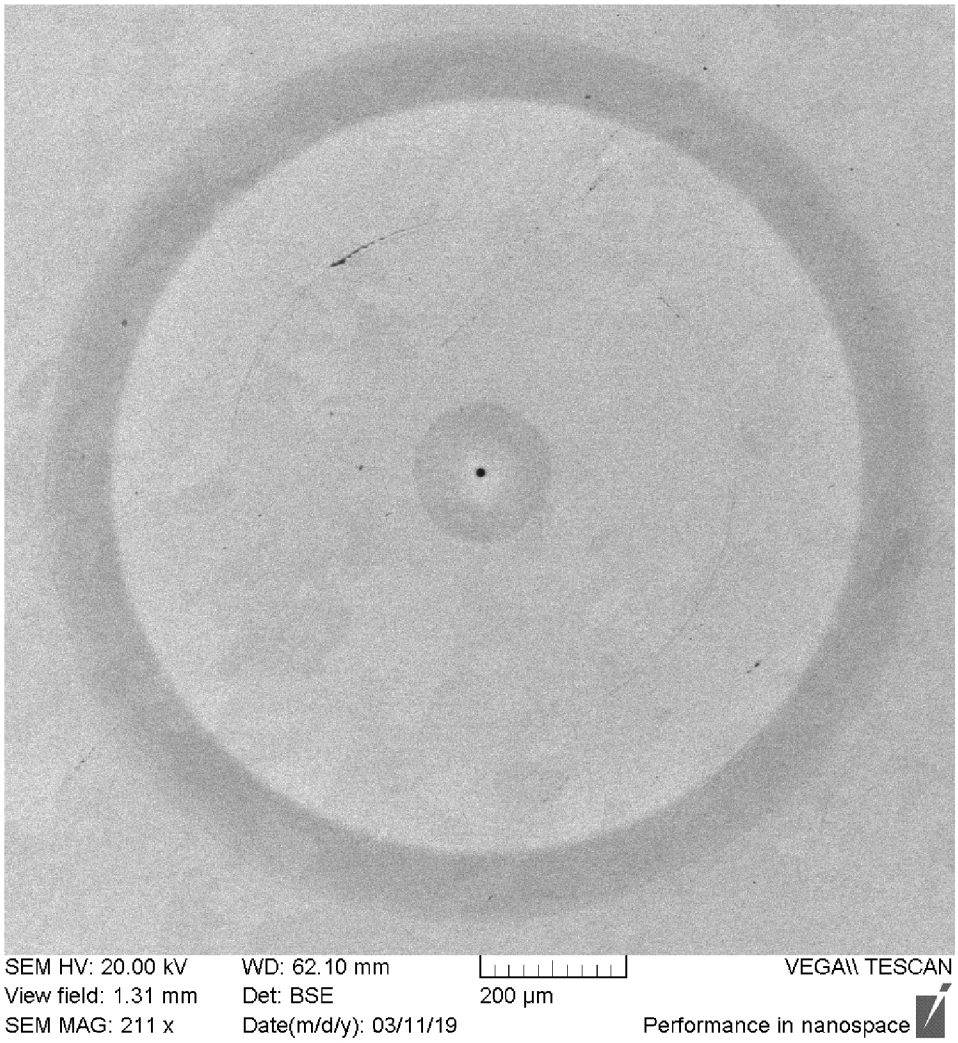


Figure C.4: Prototype image 2

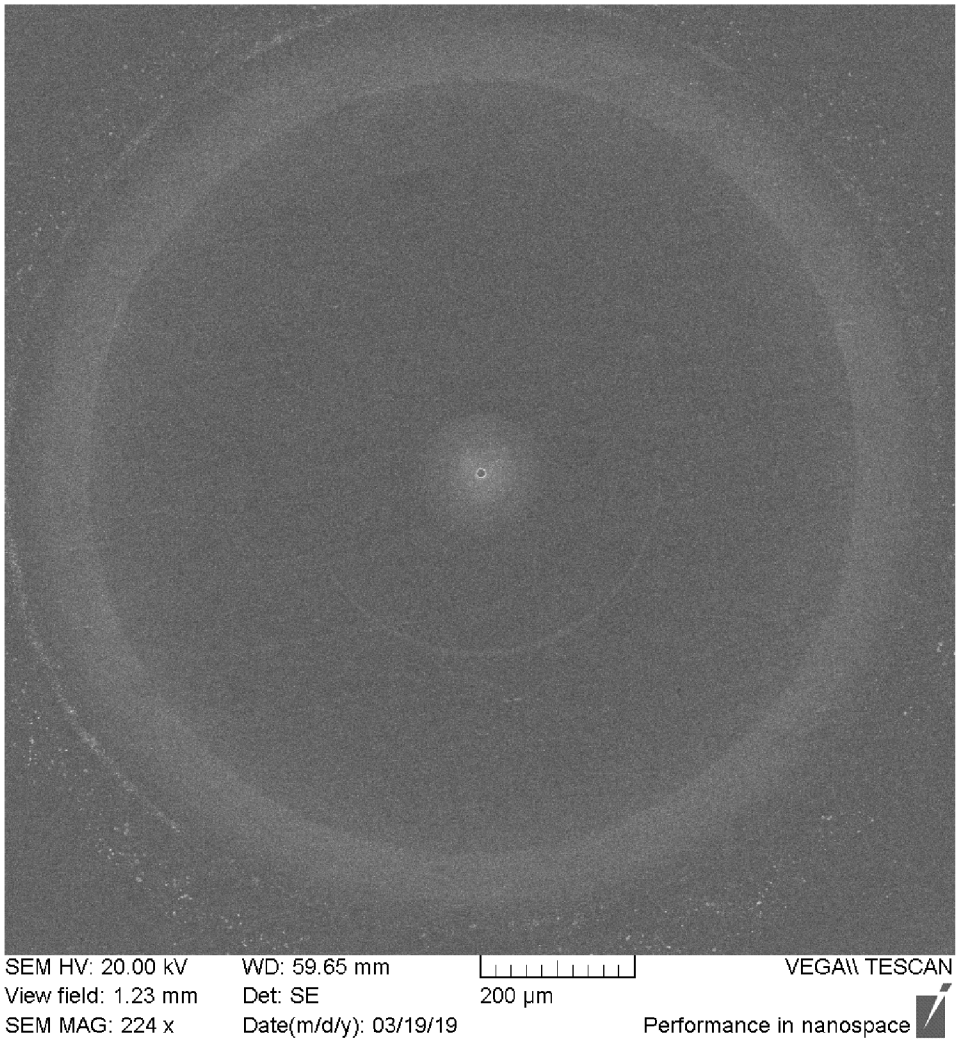


Figure C.5: Prototype image 3

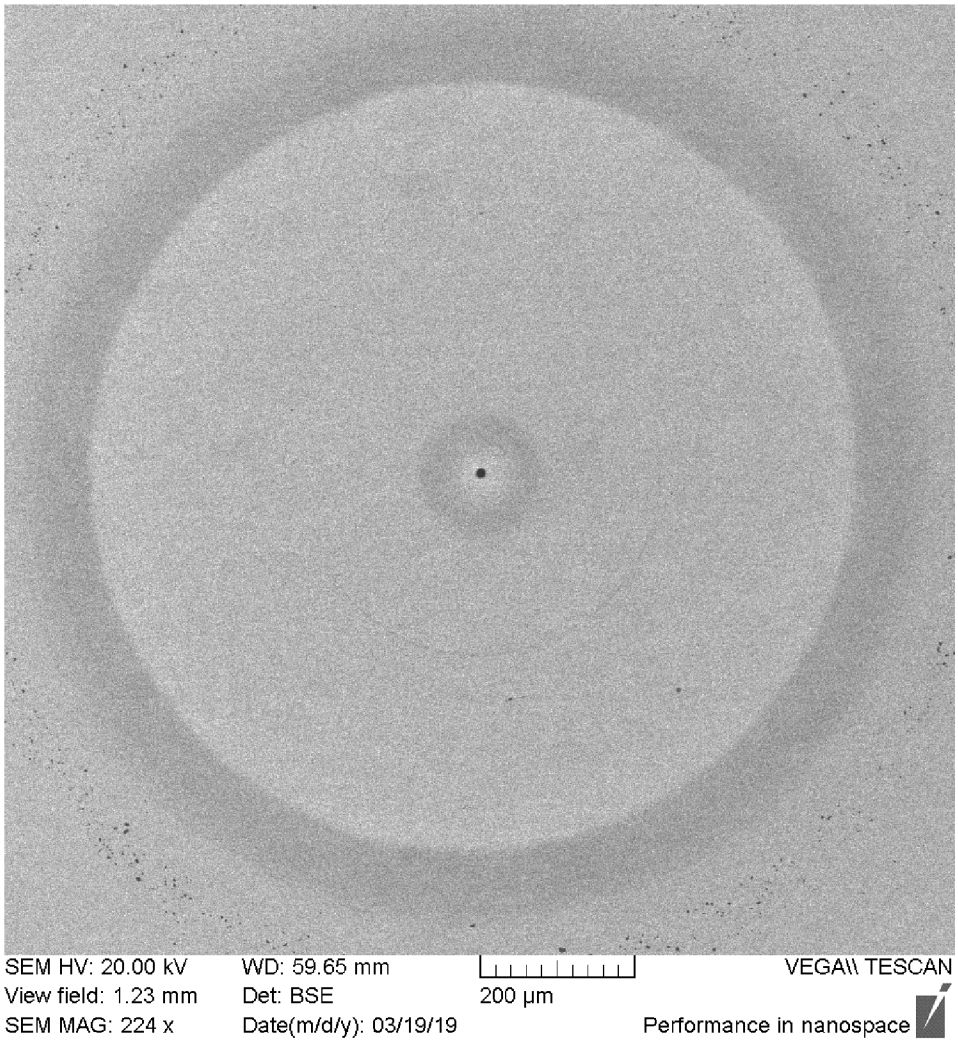


Figure C.6: Prototype image 3

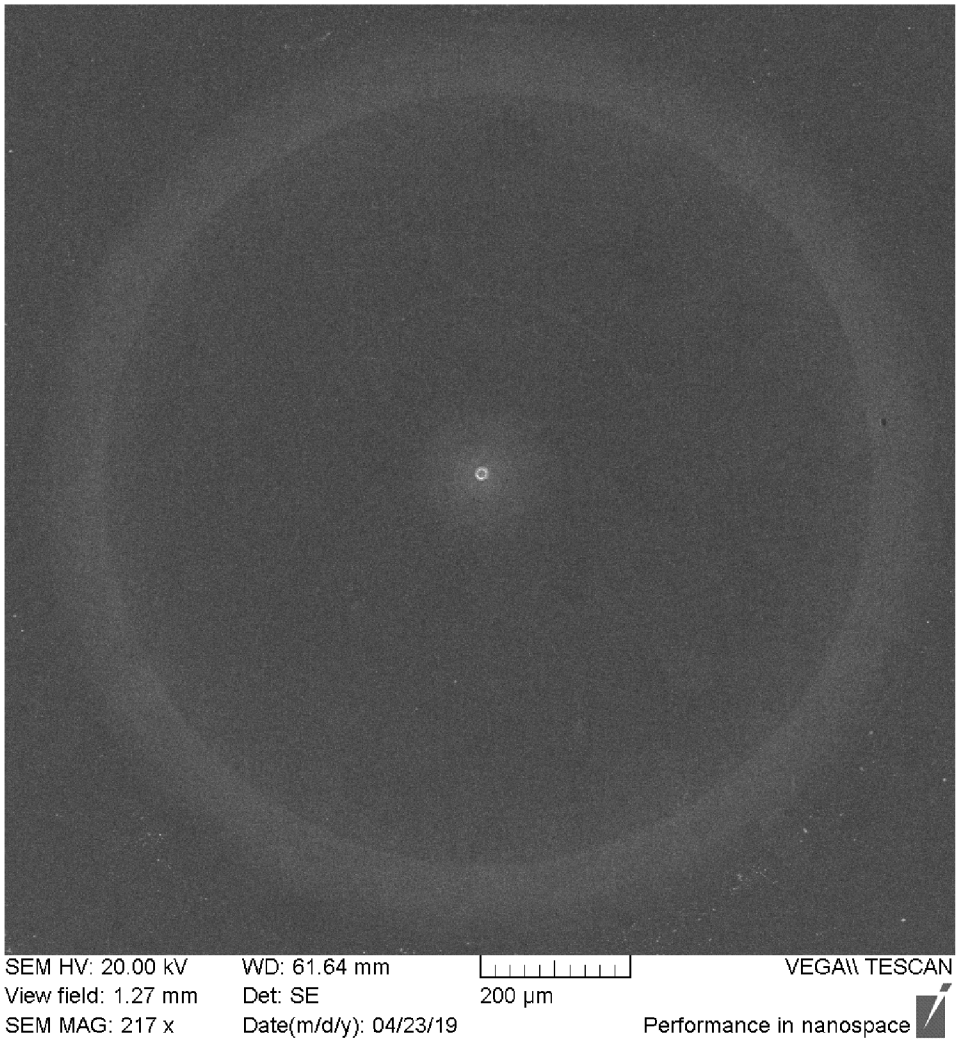


Figure C.7: Prototype image 4

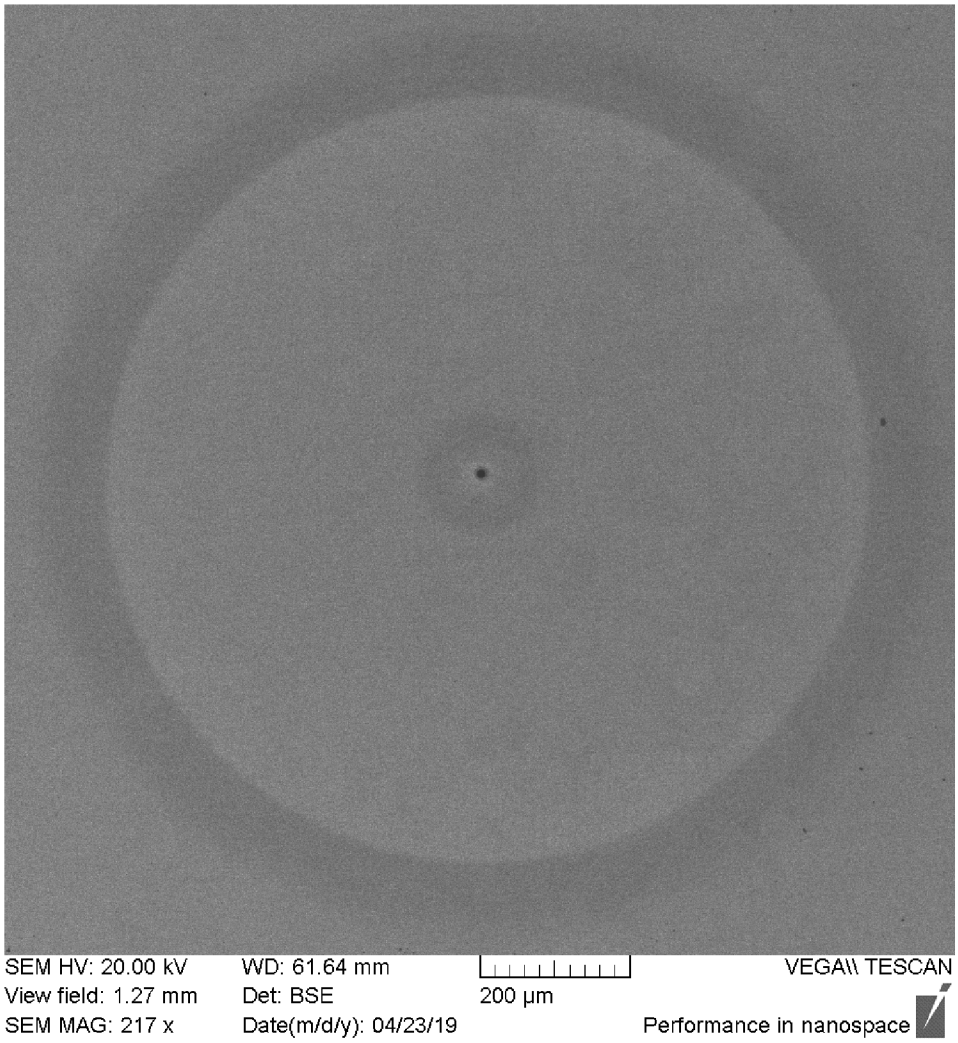


Figure C.8: Prototype image 4

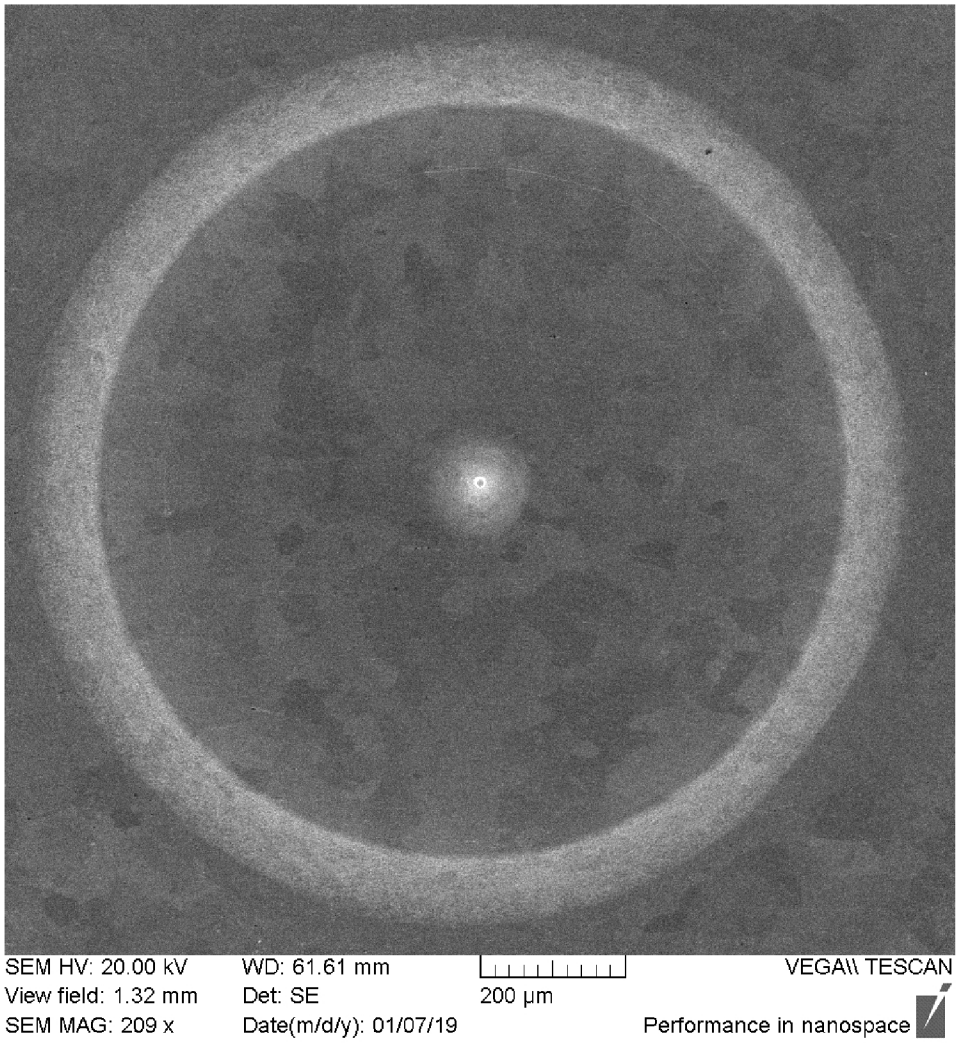


Figure C.9: Prototype image 5

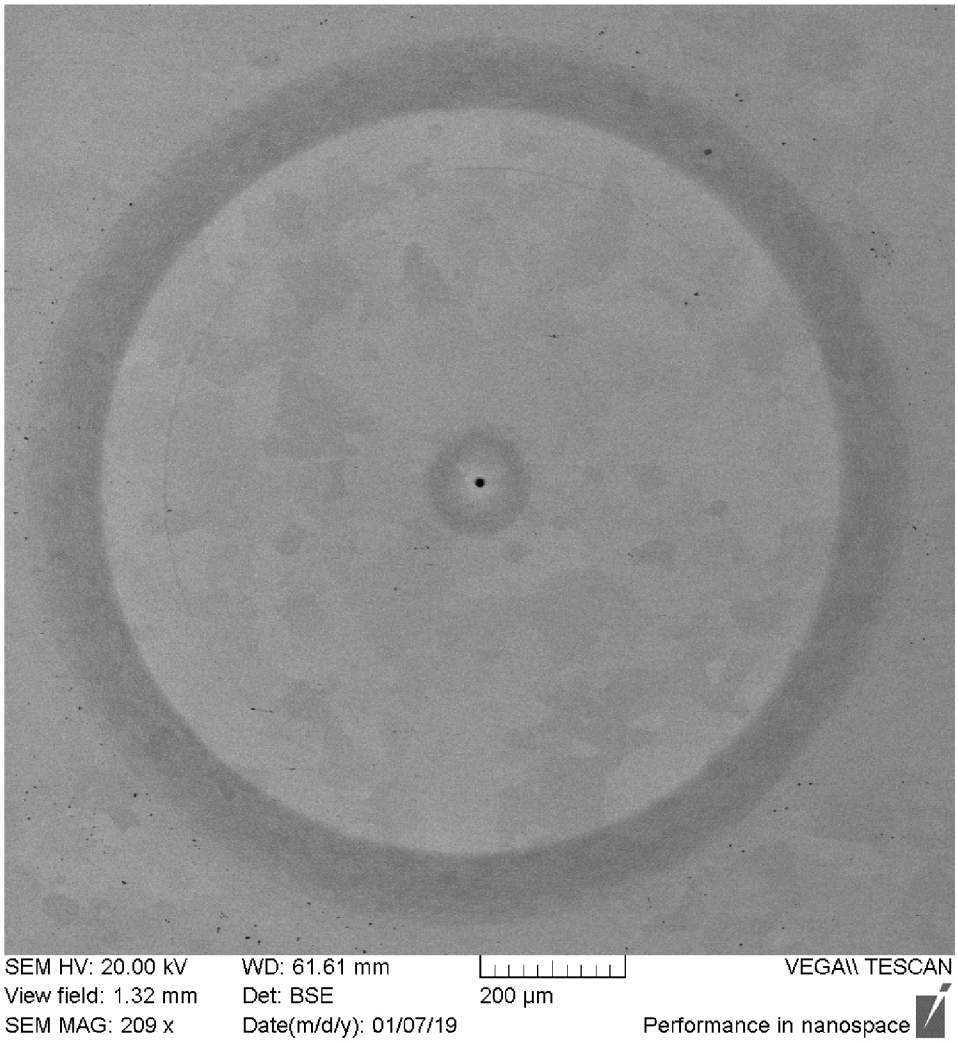


Figure C.10: Prototype image 5

D Neural networks used in the hybrid approach

Neural networks tested as the feature extractors of a hybrid model are listed here, divided by the approach they stem from, along with short description of how they differ from the base architecture, if they differ. Encoders are tested on both SE and BSE image types independently, while some DDML model branches are only tested on one of the types (used image types are provided in parentheses).

1. Encoder networks

    - **E1**. Deeper variant of the base architecture (6 encoder layers), trained for 400 epochs. (BSE or SE, 768×768)
    - **E2**. Deeper variant of the base architecture (8 encoder layers) with a lowered learning rate of 10^{-5}, trained for 600 epochs. (BSE or SE, 768×768)
    - **E3**. Deeper variant of the base architecture (8 encoder layers) with a lowered learning rate of 10^{-5} and increased Conv2D filter number from 32 to 64, trained for 400 epochs. (BSE or SE, 768×768)
    - **E4**. Deeper variant of the base architecture (8 encoder layers) with a lowered learning rate of 10^{-5} and decreased Conv2D filter number from 32 to 16, trained for 400 epochs. (BSE or SE, 768×768)
    - **E5**. Deeper variant of the base architecture (7 encoder layers) with a lowered learning rate of 10^{-4} and larger Conv2D kernel sizes in the first layers, trained for 400 epochs. See Table D.1. (BSE or SE, 768×768)
    - **E6**. Basic encoder architecture, trained for 400 epochs with decoder using Conv2DTranspose layers. (BSE or SE, 768×768)
    - **E7**. Deeper variant of the base architecture (7 encoder layers) with a lowered learning rate of 10^{-5} and larger Conv2D kernel sizes in the first layers, trained for 400 epochs with decoder using Conv2DTranspose layers. See Table D.1. (BSE or SE, 768×768)

2. Siamese network branches

   - **S1**. Base branch architecture with lowered dropout to 0.25, trained for 90 epochs. (BSE only, 768×768)
   - **S2**. Shallower version of the base architecture (3 convolutional layers) using stacked Conv2D layers, trained for 40 epochs. (BSE only, 768×768)
   - **S3**. Shallower version of the base architecture (3 convolutional layers) using stacked Conv2D layers, with final Dense layer consisting of 128 neurons, trained for 40 epochs. (SE only, 768×768)
   - **S4**. Shallower version of the base architecture (3 convolutional layers) using stacked Conv2D layers, with final Dense layer consisting of 128 neurons, without dropout, trained for 40 epochs. (BSE or SE, 384×384)

3. Triplet network branches

   - **T1**. Base branch architecture with lowered dropout to 0.2, trained for 60 epochs. (BSE or SE, 384×384)
   - **T2**. Base branch architecture without dropout, trained for 60 epochs. (BSE or SE, 384×384)
   - **T3**. Base branch architecture without dropout, with increased number of Conv2D filters and 128 neurons in the final Dense layer, trained for 60 epochs. See Table D.2. (BSE or SE, 384×384)
   - **T4**. Deeper variant of the base architecture (6 convolutional layers), without dropout, trained for 60 epochs. (BSE or SE, 384×384)
   - **T5**. Slightly deeper variant of the base architecture (5 convolutional layers), without dropout, with stacked Conv2D layers, trained for 60 epochs. (BSE or SE, 384×384)

Table D.1: Encoder architecture with decreasing kernel size.

Encoder layer number	**Conv2D kernel size**
1	7×7
2	5×5
3	5×5
4	5×5
5	3×3
6	3×3
7	3×3

Table D.2: Branch architecture with a higher number of filters.

Branch layer	**Filters**	**Neurons**
Input		
Convolutional #1	64	
Convolutional #2	128	
Convolutional #3	128	
Convolutional #4	256	
Flatten		
Dense		128