INAUGURAL – DISSERTATION

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich-Mathematischen Gesamtfakultät

der

Ruprecht–Karls–Universität

Heidelberg

vorgelegt von

Patrick Fuchs (M.Sc.)

aus Nürnberg

Datum der mündlichen Prüfung: _____

Efficient and Accurate Segmentation of Defects in Industrial CT Scans

Betreuer: Priv.-Doz. Dr. Christoph S. Garbe

Zusammenfassung

Die industrielle Computertomographie (CT) ist ein grundlegendes Werkzeug bei der zerstörungsfreien Prüfung von Leichtmetallguss- oder Kunststoffbauteilen. Eine umfassende Prüfung ist nicht nur wichtig, um die Stabilität und Lebensdauer eines Bauteils sicherzustellen. Sie erlaubt es auch, durch die Unterstützung bei der Optimierung des Gussprozesses, die Ausschussrate zu senken und, durch das Herstellen gleichwertiger aber filigranerer Strukturen, Material (und Gewicht) einzusparen. Mit Hilfe einer CT-Aufnahme ist es theoretisch möglich jeden Defekt im untersuchten Bauteil zu lokalisieren und seine Form genau zu bestimmen, was wiederum Rückschlüsse auf dessen Schädlichkeit zulässt. Jedoch ist die Datenqualität meist nicht gut genug, sodass die Defekte nicht mit einfachen filterbasierten Methoden segmentiert werden können, die direkt auf den Grauwerten arbeiten – besonders, wenn die Prüfung auf die komplette Produktion ausgeweitet wird. Im Falle der Reihenprüfung schränken die kurzen Taktzeiten die verfügbare Zeit für das Erstellen einer CT-Aufnahme weiter ein, wodurch diese rauschbehaftet und artefaktanfällig wird. Die weitreichenden Fortschritte der letzten Jahre im Bereich des Deep Learnings (und insbesondere bei faltungsbasierte neuronalen Netzen) ermöglichen es, selbst kleine Objekte in überladenen Szenen finden. Diese Verfahren bieten einen vielversprechenden Ansatz, um selbst in widrigen CT-Aufnahmen die Defekte schnell, zuverlässig und genau zu segmentieren. Der große Nachteil: Diese Verfahren benötigen eine große Menge sorgfältig annotierter Trainingsdaten, welche äußerst schwierig zu beschaffen sind – insbesondere wenn es um die Erkennung winziger Defekte in riesigen, stark artefaktbehafteten, dreidimensionalen Voxeldaten geht.

Daher beschäftigt sich ein wesentlicher Teil dieser Arbeit mit der Beschaffung präzise annotierter Trainingsdaten. Zunächst untersuchen wir, wie die manuelle Annotation von CT-Daten erleichtert werden kann: Wir erstellen eine hochqualitative CT-Aufnahme mit hoher räumlicher Auflösung und hoher Kontrastauflösung für die Annotation und übertragen die Annotationen schließlich auf eine ausgerichtete, „normale“ CT-Aufnahme des gleichen Bauteils. Diese CT-Aufnahme weist alle Herausforderungen auf, die wir im Produktionseinsatz erwarten. Die Annotationen sind dennoch uneindeutig, da sich die annotierenden Experten nicht immer einig sind, was als Defekt zu annotieren ist und was nicht. Daher untersuchen wir außerdem verschiedene Ansätze zum Erstellen künstlicher Trainingsdaten, für welche eine exakte Annotation berechnet werden kann. Präzise Annotationen sind für ein einwandfreies Training entscheidend. Wir evaluieren (i) „Domain Randomization“-Ansätze, welche mittels einfacher Transformationen eine Obermenge der Realität simulieren, (ii) generative Modelle, die dazu trainiert werden Stichproben aus der Datenverteilung der echten Welt zu produzieren und (iii) realistische Simulationen, welche die wesentlichen Gesichtspunkte echter CT-Aufnahmen abbilden. Im Zuge dessen entwickeln wir eine vollautomatisierte Simulations-Pipeline, die uns mit jeder beliebigen Menge an präzise annotierten Traningsdaten versorgt. Als erstes platzieren wir künstliche aber plausible Fehlstellen in prozedural erzeugten, virtuellen Gussbauteilen. Danach simulieren wir realistische CT-Aufnahmen, die alle typischen CT-Artefakte wie Streuung, Rauschen, Strahlaufhärtungs- und Ringartefakte aufweisen. Schließlich berechnen wir noch eine präzise Annotation für jeden Voxel aus seiner Überschneidung mit dem Fehlstellenmodell. Um zu

bestimmen, ob sich unsere realistisch simulierten CT-Daten als Trainingsdaten für maschinelle Lernverfahren eignen, vergleichen wir die Vorhersageleistung lernbasierter und nicht-lernbasierter Defekterkennungsalgorithmen sowohl auf simulierten Daten als auch auf echten CT-Aufnahmen.

Wir vergleichen unser neuartiges Deep-Learning-Verfahren mit Bildverarbeitungsmethoden und herkömmlichen Methoden des maschinellen Lernens. Diese Auswertung zeigt, inwieweit die Defekterkennung von lernbasierten Ansätzen profitiert. Die Verfahren umfassen (i) eine filterbasierte Anomaliedetektion, die Hinweise auf Defekte findet, indem sie die ursprünglichen CT-Daten von einer generierten, „defektfreien" Darstellung abzieht, (ii) eine Pixel-Klassifizierung, bei der ein Random Forest, basierend auf dicht extrahierten, manuell definierten Merkmalen, entscheidet, ob ein Bildelement Teil eines Defekts ist oder nicht, und (iii) eine Deep-Learning-Methode, die ein U-Net-ähnliches Kodierer-Dekodierer-Paar dreidimensionaler Faltungen mit einem zusätzlichen Verfeinerungsschritt kombiniert. Das Kodierer-Dekodierer-Paar liefert dabei eine hohe Trefferquote, wodurch wir selbst sehr kleine Fehlstellen erkennen können. Der Verfeinerungsschritt liefert eine hohe Genauigkeit durch das Aussortieren falsch positiver Antworten. Wir evaluieren diese Verfahren ausgiebig, sowohl auf unseren realistisch simulierten CT-Aufnahmen als auch auf echten CT-Aufnahmen, hinsichtlich ihrer „Probability of Detection", die uns verrät mit welcher Wahrscheinlichkeit ein Defekt gegebener Größe in einer CT-Aufnahme gegebener Datenqualität gefunden werden kann und hinsichtlich ihrer „Intersection over Union", die uns nähere Informationen über die allgemeine Genauigkeit der Segmentiermaske liefert. Während die lernbasierten Verfahren die Bildverarbeitungsmethode klar übertreffen, besticht die Deep-Learning-Methode besonders durch ihre Vorhersagegeschwindigkeit und ihre Vorhersageleistung auf schwierigen CT-Aufnahmen – wie sie zum Beispiel in der Reihenprüfung vorkommen.

Schließlich untersuchen wir weitere Möglichkeiten und eventuelle Einschränkungen der Kombination aus unserer vollautomatisieren Simulations-Pipeline und unserem auf Deep Learning basierenden Modell. Da die Deep-Learning-Methode selbst bei CT-Aufnahmen mit geringer Datenqualität zuverlässige Ergebnisse liefert, untersuchen wir, wie weit wir die Aufnahmezeit bei gleichzeitiger Beibehaltung korrekter Segmentierungsergebnisse reduzieren können. Weiterhin werfen wir einen Blick auf die Übertragbarkeit der vielversprechenden Ergebnisse auf CT-Aufnahmen anderer Bauteile verschiedener Materialien und verschiedener Herstellungstechniken – darunter Kunststoffspritzguss, Eisenguss, additive Verfahren und zusammengesetzte Multimaterialbauteile. Jede dieser Aufgaben bringt ihre eigenen Herausforderungen mit sich, wie ein erhöhtes Artefaktniveau oder verschiedene Arten von Defekten, die gelegentlich selbst für das menschliche Auge schwer zu erkennen sind. Wir stellen uns diesen Herausforderungen, indem wir mit unserer Simulations-Pipeline virtuelle Abbilder erstellen, die diese kniffligen Aspekte erfassen, und indem wir unser auf Deep Learning basierendes Modell auf diese zusätzlichen Trainingsdaten abstimmen. Damit können wir unseren Ansatz auf spezifische Aufgaben zuschneiden und selbst bei schwierigen Daten zuverlässige und robuste Segmentierungsergebnisse erzielen. Schließlich untersuchen wir, ob die Deep-Learning-Methode, basierend auf unseren realistisch simulierten Trainingsdaten, dazu trainiert werden kann, zwischen verschiedenen Arten von Defekten zu unterscheiden – was der ursprüngliche Grund für die Voraussetzung einer präzisen Segmentierung ist – und wir untersuchen, ob die Methode unbekannte Daten erkennen kann, für die ihre Vorhersagen weniger belastbar werden, sprich eine Unzuverlässigkeitsschätzung.

Abstract

Industrial computed tomography (CT) is an elementary tool for the non-destructive inspection of cast light-metal or plastic parts. A comprehensive testing not only helps to ensure the stability and durability of a part, it also allows reducing the rejection rate by supporting the optimization of the casting process and to save material (and weight) by producing equivalent but more filigree structures. With a CT scan it is theoretically possible to locate any defect in the part under examination and to exactly determine its shape, which in turn helps to draw conclusions about its harmfulness. However, most of the time the data quality is not good enough to allow segmenting the defects with simple filter-based methods which directly operate on the gray-values—especially when the inspection is expanded to the entire production. In such in-line inspection scenarios the tight cycle times further limit the available time for the acquisition of the CT scan, which renders them noisy and prone to various artifacts. In recent years, dramatic advances in deep learning (and convolutional neural networks in particular) made even the reliable detection of small objects in cluttered scenes possible. These methods are a promising approach to quickly yield a reliable and accurate defect segmentation even in unfavorable CT scans. The huge drawback: a lot of precisely labeled training data is required, which is utterly challenging to obtain—particularly in the case of the detection of tiny defects in huge, highly artifact-afflicted, three-dimensional voxel data sets.

Hence, a significant part of this work deals with the acquisition of precisely labeled training data. Firstly, we consider facilitating the manual labeling process: our experts annotate on high-quality CT scans with a high spatial resolution and a high contrast resolution and we then transfer these labels to an aligned "normal" CT scan of the same part, which holds all the challenging aspects we expect in production use. Nonetheless, due to the indecisiveness of the labeling experts about what to annotate as defective, the labels remain fuzzy. Thus, we additionally explore different approaches to generate artificial training data, for which a precise ground truth can be computed. We find an accurate labeling to be crucial for a proper training. We evaluate (i) domain randomization which simulates a super-set of reality with simple transformations, (ii) generative models which are trained to produce samples of the real-world data distribution, and (iii) realistic simulations which capture the essential aspects of real CT scans. Here, we develop a fully automated simulation pipeline which provides us with an arbitrary amount of precisely labeled training data. First, we procedurally generate virtual cast parts in which we place reasonable artificial casting defects. Then, we realistically simulate CT scans which include typical CT artifacts like scatter, noise, cupping, and ring artifacts. Finally, we compute a precise ground truth by determining for each voxel the overlap with the defect mesh. To determine whether our realistically simulated CT data is eligible to serve as training data for machine learning methods, we compare the prediction performance of learning-based and non-learning-based defect recognition algorithms on the simulated data and on real CT scans.

In an extensive evaluation, we compare our novel deep learning method to a baseline of image processing and traditional machine learning algorithms. This evaluation shows how much defect detection benefits from learning-based approaches. In particular, we compare (i) a filter-based anomaly

detection method which finds defect indications by subtracting the original CT data from a generated "defect-free" version, (ii) a pixel-classification method which, based on densely extracted hand-designed features, lets a random forest decide about whether an image element is part of a defect or not, and (iii) a novel deep learning method which combines a U-Net-like encoder-decoder-pair of three-dimensional convolutions with an additional refinement step. The encoder-decoder-pair yields a high recall, which allows us to detect even very small defect instances. The refinement step yields a high precision by sorting out the false positive responses. We extensively evaluate these models on our realistically simulated CT scans as well as on real CT scans in terms of their *probability of detection*, which tells us at which probability a defect of a given size can be found in a CT scan of a given quality, and their *intersection over union*, which gives us information about how precise our segmentation mask is in general. While the learning-based methods clearly outperform the image processing method, the deep learning method in particular convinces by its inference speed and its prediction performance on challenging CT scans—as they, for example, occur in in-line scenarios.

Finally, we further explore the possibilities and the limitations of the combination of our fully automated simulation pipeline and our deep learning model. With the deep learning method yielding reliable results for CT scans of low data quality, we examine by how much we can reduce the scan time while still maintaining proper segmentation results. Then, we take a look on the transferability of the promising results to CT scans of parts of different materials and different manufacturing techniques, including plastic injection molding, iron casting, additive manufacturing, and composed multi-material parts. Each of these tasks comes with its own challenges like an increased artifact-level or different types of defects which occasionally are hard to detect even for the human eye. We tackle these challenges by employing our simulation pipeline to produce virtual counterparts that capture the tricky aspects and fine-tuning the deep learning method on this additional training data. With that we can tailor our approach towards specific tasks, achieving reliable and robust segmentation results even for challenging data. Lastly, we examine if the deep learning method, based on our realistically simulated training data, can be trained to distinguish between different types of defects—the reason why we require a precise segmentation in the first place—and we examine if the deep learning method can detect out-of-distribution data where its predictions become less trustworthy, i. e. an uncertainty estimation.

Acknowledgments

During my journey through the fascinating world of computed tomography, aluminum casting, and, of course, machine learning I received a great deal of support to conduct my research and lots of motivation which particularly eased the suffering while bringing the work to paper. Therefore, I would like to take the chance to express my deep gratitude:

1. I would like to thank all those who made this thesis possible: Christoph Garbe who willingly accepted the supervision of the work and stopped by even after the end of his work day. I also would like to thank him as well as Fred Hamprecht for providing the chance for my doctoral studies in computer science at the Faculty of Mathematics and Computer Science, Heidelberg University. Tobias Dierig who arranged the cooperation with Volume Graphics (VG) and continued providing guidance even after he moved on to another company and Thomas Günther and the board of directors of VG for offering me the position as "PhD Candidate" and granting me the freedom to carry out my work independently.

1. I would like to thank Thorben Kröger, Sven Gondrom-Linke, and Jonathan Hess for their intensive support, providing me with practical advice and with plenty of data, helping me with the publication of research papers in application conferences and scientific journals as well as for proof reading this work.

1. I would like to thank our CT experts for clicking through tons and tons of voxel data, minutely labeling the casting defects in the CT scans. Furthermore, I would like to thank all the institutes and cooperation partners who do not wish to be mentioned by name for providing further CT scans of real parts and Daniela Handl for organizing the pairs of high-quality and "normal" CT scans of the cast aluminum parts.

1. I would like to thank Sören Schüller, Matthias Fleßner, and Benjamin Bertram for all the helpful contributions and fruitful discussions, Benjamin Maier for helping me to raise the simulation pipeline to a production level, my colleagues of the material analysis team and the former image processing team who protected me from all the rebounds originating from my time as a working student at VG, the IT department for fulfilling all my special demands and requests, as well as all my other colleagues at VG for their openhearted support and the pleasant working atmosphere.

1. Last but not least, I would like to thank my family, who kindly accepted that I spent my scarce visits working on the thesis, my friends, who never got tired of inviting me over, and in particular Anna-Maria, who bore all my whining, read through all the unpolished machine learning and computed tomography stuff, and—what I'm most grateful for, even though I didn't always show it immediately—allowed me to rest my mind and managed to cheer me up with all the little distractions.

I hope you all enjoy reading!

Contents

List of Figures

List of Acronyms

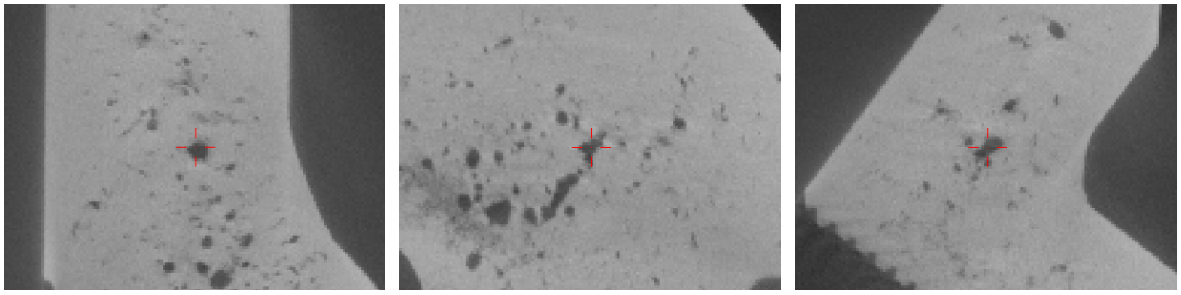
AM	additive manufacturing	118
AUC	area under curve	106
CAD	computer-aided design	48
CDF	contrast discrimination function	96
CDD	contrast detail dose	96
CNN	convolutional neural network	25
CNR	contrast-to-noise ratio	38
CRF	conditional random field	30
CT	computed tomography	1
DoG	difference of Gaussians	31
ESD	equivalent sphere diameter	94
Faster R-CNN	faster region-based CNN	29
FBP	filtered back-projection	11
FCN	fully convolutional network	25
GLCM	gray-level co-occurrence matrix	69
GPU	graphics processing unit	12
GAN	generative adversarial network	44
iIoU	instance-based intersection over union	103
IoU	intersection over union	91
MAE	mean absolute error	84
MDA	mean decrease in accuracy	71
MDI	mean decrease in impurity	71
MSE	mean squared error	83
MTF	modulation transfer function	96
NDT	non-destructive testing	1
OOD	out-of-distribution	126
POD	probability of detection	91
PR curve	precision recall curve	91
ReLU	rectified linear unit	23
ROC	receiver operating characteristics	91
ROI	region of interest	6
RPN	region-proposal network	29

SGD	stochastic gradient descent	23
SNR	signal-to-noise ratio	11
SSIM	structural similarity	83
YOLO	you only look once	29

1. Introduction

In the automotive and aerospace industries, light metal cast parts are widely used due to their beneficial material properties: they are lightweight yet stable. The reduced weight allows resources such as fuel to be conserved and the stability holds up a high level of durability and safety. However, properly producing cast parts is not an easy task and usually the fabricated parts are never free of anomalies like voids or other defects. Due to reactions with the sand of the mold or the surrounding air while pouring the liquid metal and the material contraction during the subsequent cooling and solidification process, various types of defects like gas pores, shrinkage cavities, and micro cracks emerge [1–4]. Such defects reduce the stability of the part. A possible compensation would be making critical regions more massive. Unfortunately, this often violates design constraints and quickly negates the advantages of the reduced weight of light metal parts. But not all types of defects contribute equally to material fatigue and, thus, are not equally harmful [1]. To ensure the durability and reliability of the produced parts, it is necessary to ensure that only the less harmful defect types occur and that these defects are located only in non-critical regions of the part. We can achieve this either by using destructive methods, e. g. by cutting the part open and examining the position and size of the defects in the cut plane, i. e. metallography [5], or by using methods of non-destructive testing (NDT) like X-ray computed tomography (CT) [6] which allows us to reconstruct a view of the inner parts of an object. Metallography has two major drawbacks: First, the object under examination will definitively be destroyed and cannot be used afterwards. This means the production process needs to be stable enough to constantly yield a similar output. Second, the results like the number of defects and their size heavily depend on where we cut the object apart [1]. In return, we obtain a high spatial and contrast resolution. NDT has the benefit of leaving the object under examination unscathed and assuring that a specific part has the desired properties regarding reliability and durability. Furthermore, we can precisely determine the three-dimensional size, shape, and position of each defect instance, which is necessary to determine their harmfulness [2]. Containing all this information, the CT scan can be seen as a digital twin [7] of a specific object and, thus, can further be beneficial for other tasks like metrology, simulations, or **cae!**. NDT methods, in consequence, enable us to save material on the part by creating thinner structures and to reduce the rejection rate while guaranteeing stability and safety. In exchange, we have to deal with a reduced spatial and contrast resolution as well as a higher artifact-affliction.

Checking an industrial CT scan for defects is a cumbersome task. A CT scan is a three-dimensional gray-value image. As we do not only have surface information but also details about the inner life of an object, a three-dimensional rendering is not capable of displaying all the information. Therefore, a CT scan needs to be inspected layer by layer, i. e. two-dimensional slices, looking for deviations in the gray-values. In Figure 1.1 we show three examples of axis-aligned slices centered around the same image point in the three-dimensional CT scan, i. e. the same voxel. For a human it is hard to grasp the three-dimensional shape of a defect by only looking at those two-dimensional slice-views. Thus, it is very hard to determine which portions are connected and which are separate instances. Because of that, a three-dimensional rendering of a defect requires precise information about its



(a) top view (b) right view (c) front view

Figure 1.1.: The three axis-aligned slices of a CT scan of a cast aluminum part are centered around the same voxel which belongs to a shrinkage cavity defect (red cross). By going through the slice views, it is hard to determine which of these defects are individual instances and which are actually connected.

boundary, i. e. a precise segmentation. Every voxel that belongs to a defect needs to be labeled as such. This again requires a consistent depiction of the gray-values. Accordingly, a proper human inspection requires standardized, calibrated monitors. While it already is quite hard to distinguish between defective and flawless voxels—especially for smaller occurrences—image artifacts further impede the inspection process and lead to a rapid exhaustion of the quality experts. Nevertheless, the inspection of each and every part of the production is beneficial: Ruling out defective parts in early stages of the production offers the opportunity to save time and money [3], for example, by avoiding expensive post-processing steps and making sure the final product works reliably. However, the accompanying complications raise the need for an automated inspection.

Before we start automating the inspection, we need to know what we are looking for. Hence, we provide a brief overview of the different types of defects that typically occur in light metal castings describing their origin and their appearance [1, 3, 8, 9]:

Gas pores There are several reasons for the emergence of gas pores [3]: (i) Gases which are dissolved in the liquid melt precipitate during the cooling process because with decreasing temperature the solubility of the melt is reduced. If the gas cannot escape the material, bubbles of varying size manifest in the part. (ii) When pouring the melt into the mold turbulence can occur with the air in the mold mixing in bubbles of air. (iii) The melt further reacts with the water in the green sand of the mold, which is necessary to keep the included clay moist and stable, forming hydrogen and metal oxides which are trapped in the material [8]. Gas pores typically are spherical voids with a smooth and clean surface [1].

Shrinkage cavities In general the liquid melt has a higher volume than the solid metal alloy. The contraction of the material during the cooling and solidification process can lead to a rupture of the material structure opening a void in the object. Shrinkage cavities have a rough, irregular surface and an arbitrary, branched shape [1].

Solidification cracks If the stress which is built up during the cooling and solidification process exceeds a certain limit, the material separates. This type of defect occurs only at the end of the solidification process [10]. In the CT scan they appear as flat, sharp, and jagged discontinuities.

Structural loosening Irregularities in the structure of the material with less desirable properties are described as structural loosening. An example is the occurrence of large sponge-like clus-

ters of tiny pores which lie close together. Because these pores cannot be resolved individually in the CT scan, they occur as a darker region within the material.

Inclusions of foreign material Besides the previously mentioned oxide films several non-metallic inclusions impede the structure of a cast part, for example, due to impurities in the melt. Furthermore, parts of the mold can be dragged along during casting, forming sand inclusions in the solidified product.

Besides that, geometrical defects have to be considered [11]: Pouring the molten metal too quickly or under too high pressure can lead to a deformation of the mold. In contrast, pouring the molten metal too slowly or with too little pressure can lead to an early solidification, i. e. a cold run or cold shot. Furthermore, if the individual parts of the mold do not flush, the molten metal can escape the form at their boundary and lead to flashes or fins. For the purpose of this work we limit ourselves to the detection of gas pores, shrinkage cavities, and cracks in combination with regional occurrences of structural loosening and put a special focus on cast aluminum.

For an automated inspection, a precise segmentation is necessary in order to determine properties like the sphericity of a defect, its volume, the size of its surface, the distance to the material boundary, or the distance to other defects. These parameters then allow for a subsequent categorization of the found defects and a determination of their harmfulness. Furthermore, the domain experts operating the defect detection methods do not want to need to tune some model-related parameters for each and every data set. They just want to define what makes a defect critical and when a defect is still acceptable. Machine learning methods offer this convenience: at least at inference-time there are no parameters to tune. Instead, they require a training phase to set the right parameters beforehand. Currently, deep learning methods are reported to improve the state-of-the-art in almost all fields of machine learning. Of particular interest for this work are the successes of deep learning in semantic segmentation [12–14], scene understanding [15–17], and especially the processing of medical data [18–22]. Sometimes, the presented methods surpass even a human-level of performance [23–25]. Hence, deep learning represents a promising approach to support the detection and segmentation of defects even in challenging CT scans of cast aluminum parts. However, before we establish a novel approach for the segmentation of defects based on deep learning we need to verify that this approach really outperforms traditional methods in terms of precision and prediction time [26]. Moreover, a lot of labeled training data is necessary to train a deep learning model [27], which, however, is quite hard to obtain. Labeling objects of everyday life in images already is quite hard [28]. The annotation of defects in CT scans additionally requires the knowledge of domain experts which further increases the cost of generating a satisfying training set. These challenges are also known from the labeling of medical CT scans, where we encounter similar problems [29]. Hence, we turn to realistic simulations to provide us with enough training data. While conducting the research for the approach to train a network only on synthetic data, this approach became more and more popular and was applied on a grand scale in other fields, too, e. g. Microsoft's "Synthetic Data with Digital Humans" or Unity's "SynthDet" which is inspired by [30].

As this work focuses on the development of a new field of application of deep learning methods, the main contributions are the following:

- A way to obtain enough training data with a precise ground truth such that deep learning models can be trained to reliable detect and accurately segment defects. This is done by realistically simulating CT scans of cast aluminum parts.

- A comprehensive comparison of reference-free defect detection methods, based on both machine learning approaches and image processing techniques.
- An extensive evaluation of how deep learning methods help to improve the detection of defects in CT scans of cast aluminum and other parts.

Before diving deeper into the data, the methods, and their evaluation, we start off with a brief introduction into industrial computed tomography in Chapter 2 to provide an idea of the target domain and touch upon the challenges associated with the inspection, i. e. the artifact load of the data and the timing constraints for the inspection. Chapter 2 also addresses the basics of semantic segmentation algorithms and in particular deals with the concepts of deep learning approaches. Furthermore, we discuss the state-of-the-art of defect detection methods considering semi-automated approaches which guide a human inspector as well as fully automated approaches for specific parts which, however, rely on a defect-free reference specimen.

Then, the first issue which we tackle on our journey to a reference-free deep semantic segmentation of defects in CT scans is the acquisition of the necessary amount of training (and validation) data in Chapter 3. A machine learning algorithm can only be as good as the data set used for training. For this reason, we examine how we can improve the labeling process for industrial CT data and evaluate different approaches to create synthetic data, for which we can compute a precise ground truth. We end Chapter 3 with carefully designing a simulation pipeline to provide an infinite source of precisely labeled realistic training data for all sorts of CT-related NDT-tasks.

In Chapter 4 we present the state-of-the-art methods in terms of reference-free defect detection, i. e. methods that do not rely on a specific defect-free specimen. We present an image processing approach, which does not use any machine learning techniques and develop a method that comprises a traditional machine learning approach. These approaches form a sound baseline which we use for an exhaustive comparison. At the end, we introduce our deep learning model.

A healthy mistrust is in the nature of a quality expert. Therefore, we need to thoroughly evaluate our deep learning model. We provide a comprehensive quantitative evaluation on simulated and real CT scans comparing our deep learning model to the traditional approaches in Chapter 5. For this we rely on two measures: one mostly used in the application domain, i. e. the probability of detection, and one familiar to the image processing domain, i. e. the intersection over union. Furthermore, we compare to results taken from other inspection techniques (namely metallography).

Finally, we discuss the vast corridor of possibilities that opens up by combining realistically simulated training data and deep learning models and tackle the most common concerns of domain experts regarding the reliability of those models in Chapter 6. We investigate the use for in-line inspection, explore how we can adapt our model to different tasks, and examine its prediction confidence on out-of-distribution data.

Parts of this thesis were previously presented in [31,32]. Nevertheless, this thesis adds further detail to the presented topics and additionally deals with (i) the adaptation to different types of defects, i. e. fatigue cracks, (ii) the estimation of the confidence of the deep learning model (or rather its uncertainty on novel data), (iii) the correlation of the probability of detection with other quality measures of CT scans, and (iv) the classification of defects telling apart gas pores from shrinkage cavities.

2. Theoretical Background and Related Work

This work involves a broad variety of scientific fields: We deal with light metal casting and appropriate inspection techniques, computed tomography as imaging domain, image processing and machine learning for automation, simulation techniques in combination with procedural modeling for data generation, and more. Having introduced the problem of detecting defects in the introduction (Chapter 1), we leave it to the reader to further enter the field of light metal casting. In this chapter we focus on the most essential theoretical aspects relating to the input domain, i. e. computed tomography, the state-of-the-art in defect recognition, and the exploration of semantic segmentation methods. Besides knowing about the actual task, a good understanding of the input domain is crucial for developing a reliable machine learning model. Hence, we directly start off with a short introduction to CT: We set up the terminology which is used throughout this work, we explain the theoretical aspects of a CT system which are necessary to understand the simulation process, and we discuss the image artifacts that come along with CT and impede an automated inspection. Finally, we discuss what makes a good CT scan and how the necessary efforts conflict with the challenges of industrial applications—and in-line scenarios in particular. Then, we briefly discuss the state-of-the-art defect recognition methods, which are currently used to inspect CT scans of cast aluminum parts: ranging from a guided manual inspection of individual parts to the automated in-line inspection of a production line. Here, we explain the benefits and drawbacks of the currently used image processing and anomaly detection methods, describing the improvements that were most beneficial for the inspection process. Finally, we provide helpful insights to the field of machine learning with a focus on the deep learning methods which, as of late, are reported to outperform traditional methods. We again start off with an explanation of the terminology common to the field of machine learning and semantic segmentation in particular. Then we explore the data-driven end-to-end trainable deep learning methods for semantic segmentation and compare them to traditional approaches that consist of individual steps for candidate selection, feature extraction, and classification, which have to be tuned separately. Other, minor theoretical aspects will be explained in-place when necessary.

2.1. Computed Tomography

If we would like to simply have a look into an object, we would not need an expensive computed tomography, which comprises plenty of projections taken from different angles. We could use a simple X-ray radiograph, i. e. a single projection. The great benefit of CT, in contrast, is that we end up with a *digital representation* of an object with the full volumetric information, i. e. we can precisely tell where a defect is located and how it is shaped, instead of just knowing that there is a defect in the part under examination. In addition, we can use the depth information of a CT scan to determine whether the part matches its model and whether we would open a defect during post-processing, as well as to conduct further simulations, for example, regarding the stability of the part.

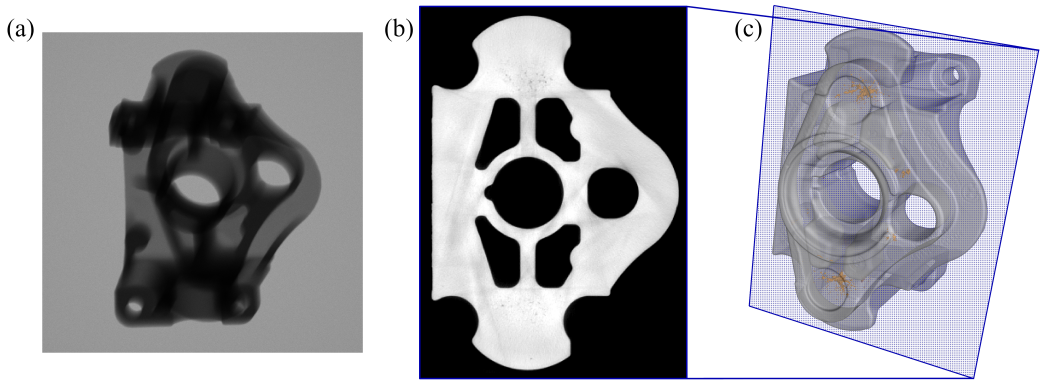


Figure 2.1.: The CT scan which consists of many volumetric data points, i. e. voxels, arranged in a regular grid is referred to as volume (c). A single layer of this volume is called slice (b). If not stated otherwise, we assume the slices to be axis-aligned. The slices may not be confused with the projections from which the CT scan is reconstructed, i. e. the radiographs (a).

As every domain, computed tomography has its own terminology (see Figure 2.1). Therefore, we briefly explain a few terms that we use throughout this work first:

Voxel CT scans produce three-dimensional image data. Corresponding to the pixel in an image, the data points in three-dimensional data are called *voxel*. Most commonly, the data is arranged in a regular grid. This impliactes that each voxel has a spatial extent, representing the gray-value within a cell of this grid. Voxels do not necessarily have to be cubical. They can also be cuboid. For instance, in medical data we often encounter a primary scan direction [33,34], usually the direction in which the patient is moved through the CT system. The reconstructed image then consists of voxels which have a longer side aligned with the scan direction. However, throughout this work we only consider isotropic (cubical) voxels.

Slice and Slab A two-dimensional set of connected voxels taken from the three-dimensional CT scan is called *slice*. The slice is a part of the reconstructed CT scan and, thus, may not be confused with a projection. Due to its two-dimensional nature a slice can be rendered easily on the screen. With the CT scan basically being a stack of slices, we can browse through a CT scan layer-by-layer—or better: slice-by-slice. Throughout this work, when talking about slices, we mean axis-aligned slices. While a slice can be drawn at any arbitrary alignment, an interpolation is involved, as long as they are not axis aligned. The combination of multiple adjacent slices is referred to as *slab* (or thick slab).

Region of Interest An arbitrary selection of voxels is called a *region of interest (ROI)*. While a ROI can contain any combination of voxels, it usually refers to a cuboid set of adjacent voxels. The voxels within a ROI are typically selected according to a semantic meaning. For example, a ROI can be created to frame critical connections within a part or simply separate the part from the background.

Volume The complete set of voxels within the gray-value-grid, i. e. the data of the CT scan as a whole, is referred to as *volume*. In contrast to polygon meshes, which are also three-dimensional data representing a surface or material boundary, in a volume we have further

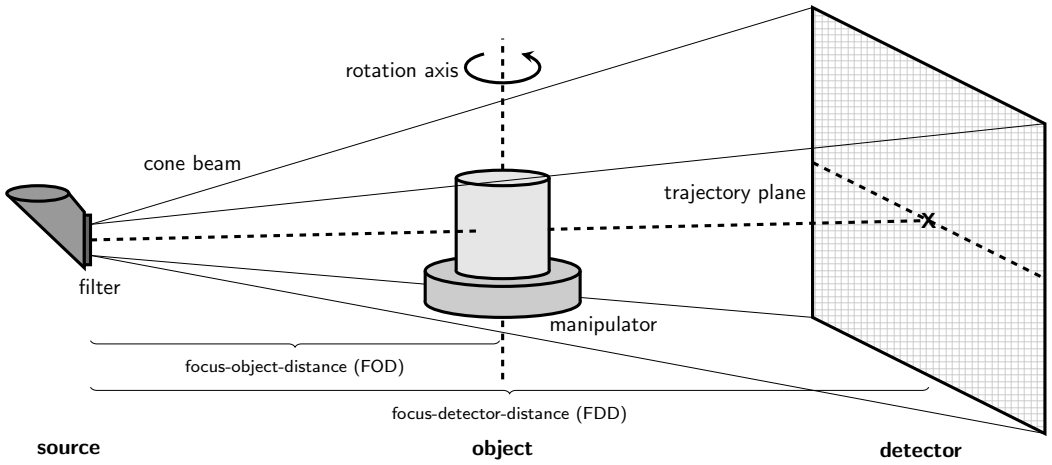


Figure 2.2.: The principal setup of an industrial (cone beam) CT system. The source is usually fixed at one side of the system. The detector, opposite of the source, is movable so that the focus-detector-distance can be adjusted. In between source and detector the part under examination (object) is mounted on a manipulator (the rotary plate) which rotates the object during the scan. The manipulator is movable as well so that the focus-object-distance is adjustable, too.

information about the inner structure of a part. This additional data makes it hard to render all the information on a two-dimensional screen at once.

Gray-value Each voxel contains a value, representing the average linear attenuation coefficient at a given point in space. As this value is usually depicted in gray-scale on the screen, it is called a *gray-value*.

This shall not be another CT-textbook. Therefore, in this section, we only explain what is necessary for a realistic simulation of CT scans, present the (almost) inevitable image artifacts that impede an automated inspection, and provide a brief guide on how to prepare a high-quality CT scan while explaining why timing constraints render such CT scans impossible—particularly in in-line inspection scenarios. For further details we refer to [33–35]. If not stated otherwise, we adhere to [33] throughout this chapter.

2.1.1. Setup of a CT System

The typical setup of a CT system comprises an X-ray source to produce the necessary radiation, a detector to measure the attenuated X-ray spectrum after penetrating an object, and a rotary plate on which the part under examination is mounted. All components are mounted on a solid platform to minimize influences of the environment—each subtle vibration can introduce further artifacts to the final image. In industrial CT we usually deal with much higher energies than in medical CT to acquire the projections for a CT scan. Thus, the CT system has to be placed in a special radiation protection chamber. Figure 2.2 shows an outline of a typical system for *cone beam CT* as it is used, for example, in quality laboratories. The individual components are presented in more detail below.

The Source. The source produces X-ray radiation by accelerating free electrons and shooting them towards a target anode. The free electrons originate from a heated coil at the cathode. The anode in a typical source is usually a tungsten coated copper element, which additionally disperses the emerging heat. Two processes in the anode are involved in producing the X-ray radiation: Firstly, there is the bremsstrahlung. When penetrating the dense material of the target anode, various processes decelerate the accelerated electrons (e^-). For example, when it comes to the interaction with the nucleus of an atom in the target material, Coulomb forces change the direction and velocity of the e^-. The differences in energy are converted into heat and bremsstrahlung. The bremsstrahlung creates a continuous spectrum of different wave lengths (see Figure 2.3a). Secondly, if an electron e^- collides with another electron e_K^- of an atom in the material of the target anode, the e_K^- is shot out of its place in the inner shell K of that atom. Since both electrons leave the atom, the empty space is immediately filled by an electron e_L^- of an outer shell L. As the inner places are energetically more favorable this process releases energy. The resulting difference in energy is converted into X-rays of a characteristic wave length. This portion of the radiation is responsible for the characteristic peaks K_α and K_β in the polychromatic spectrum (see Figure 2.3a). Where K represents the shell with the empty slot and α represents the shell from which the replacing electron originates, i. e. the next shell (L). Accordingly, $beta$ corresponds to the M shell.

The important parameter here is the acceleration voltage, which defines the minimum wave length of the polychromatic spectrum. The current further changes the number of free electrons which are accelerated. Thus, these parameters define the *contrast resolution*. In plain radiography we can focus on an area of interest and tune the current to obtain the best possible contrast for the penetration length in this area. Other areas of different penetration lengths are either over-exposed or under-exposed. In CT, however, we need to obtain at least some contrast for all penetration lengths. Thus, it is better to increase the voltage and introduce photons of higher energies instead of just increasing the number of photons (refer to the following paragraph "The Object."). The full contrast is restored due to the combination of multiple projections of different angles during reconstruction. Note that the X-ray photons emerge from the full penetration length of the electrons in the target material, not just from the point where they meet the anode. That means even for a low voltage and current, i. e. at low power, where the electron beam is more focused the effective focal spot is still larger because of the penetration depth of the electrons in the anode. Hence, the focal spot is an important factor limiting the possible *spatial resolution*.

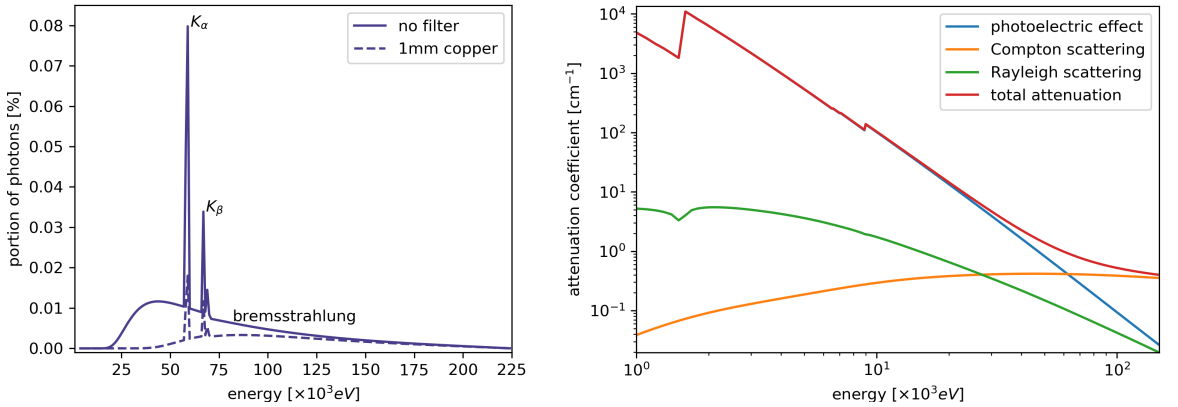
The Object. Between source and detector, we place the part under examination, i. e. the object. While it is typical for medical CT that source and detector evolve on a fixed path around the patient, in industrial CT the object is placed on a rotary plate and source and detector are kept at fixed positions. This allows to vary the distances between source and object, object and detector, as well as between source and detector and with that the magnification of the object. If the object is closer to the source, we have a higher magnification (with a magnification factor $m > 2$). Here, the size of the focal spot limits the *spatial resolution*. If the object is closer to the detector, we have less magnification (with a magnification factor $1 < m < 2$). Here, the size of the image elements of the detector limits the spatial resolution. Furthermore, when deciding about the magnification we have to consider the absolute distances. With the *distance square law* telling us we only obtain a quarter of the intensity at the detector when doubling the distance, we in consequence would lose *contrast resolution*.

When penetrating the object, the X-rays interact with the material of the part and are attenuated by several physical effects. Most important are Rayleigh scattering, the photoelectric effect, and Compton scattering. For energies above $1.022 \cdot 10^6\ eV$ there further is the effect of pair production. However, as we deal with standard CT we stay below that limit throughout this work. Rayleigh scattering describes the process of a photon, which has not enough energy to free an electron, interacting with the atoms in the material of the object. These photons are just deflected without a change in their energy. The photoelectric effect describes the process of a photon passing its entire energy to an electron of an atom in the material of the object. The energy of the photon is completely absorbed and helps an electron of an inner shell to escape its bound state. Compton scattering describes the process of a photon interacting with a valence electron of an atom in the material of the object. These electrons are only weakly bound and do not need the entire energy of the photon. Hence, the photon is deflected and moves on with a lower energy after helping the electron to escape its bound state. For lower energies the total attenuation is dominated by the photoelectric effect. When moving to higher energies the photons are more likely to pass the material or to be subject to Compton scattering (see Figure 2.3b). Note that different materials have different absorption properties, inducing characteristic peaks in the attenuation depending, for example, on the energy that is necessary to disentangle an electron of an inner shell. Moreover, denser materials have a higher attenuation as the probability of an interaction is increased. Thus, the attenuation not only depends on the penetration length s (the thickness of the material) but also on the energy E of the photons and the properties of the material, i. e. its atomic number and its density. The intensity I which is measurable at the detector is then defined according to the *Lambert-Beer law* (see Equation (2.1)), where I_0 is the initial energy emitted by the source and μ the attenuation coefficient.

$$I = I_0 \cdot e^{-\int_E \int_s \mu(s,E) ds\, dE} \tag{2.1}$$

To obtain the best possible *contrast resolution* we, therefore, need to ensure penetration lengths which are as uniform as possible so that we can adjust the voltage of the source to penetrate the longest side of the object just enough to avoid underexposure. That means that elongated objects, for example, should be mounted upright on the rotary plate. Due to the higher energies the contrast resolution of the individual projections often is less desirable, but this effect is mitigated by the high number of projections used to create a CT scan.

The Detector. Opposite of the source, the detector measures the remaining X-ray radiation. Flat panel detectors which consist of three layers are typical in industrial CT: The sensors which comprise a photo diode and thin film transistor for converting visible light into an electronic signal are printed on a glass substrate as carrier material via thin film coating. On top of the electronics layer there is a scintillating layer which converts the X-rays into visible light. Important properties of the scintillating layer are its conversion rate, i. e. how many X-ray photons are converted into how many photons of visible light, and its decay time, i. e. how long does it take until no more photons of visible light are emitted after the last incident of a X-ray photon. Especially the latter is important for CT because we have to make hundreds of consecutive projections in a short period of time. During exposure, the photo diodes turn the visible light of the scintillating layer into an electric charge which is proportional to the incoming amount of X-rays. The thin film transistors then start



(a) Spectrum of a tungsten target at 225 kV with the characteristic peaks K_α and K_β. The filter removes 74 % of the photons in total.

(b) The attenuation coefficient of the aluminum alloy EN AW 2014 [36] as function of the energy of the photons. The two peaks in the attenuation are characteristic for aluminum and copper.

Figure 2.3.: With the information about the polychromatic spectrum of the X-ray source and the attenuation in the object, we can compute the intensity that arrives at the detector.

the reading process, passing the charge on to the read-out electronics which amplify the signal and convert it to a digital representation.

The read-out electronics need to be as low-noise as possible so that subtle differences can be measured. Nevertheless, it is best to operate the detector in a way that about the lowest ten percent of the possible value range are not used. These values typically vanish in the static electronic noise of the device [37]. In addition, it is important to avoid an overexposure. At this point there would be no measurable differences anymore as further X-rays arrive at the detector. Furthermore, the quantization that takes places when digitizing the analog signal limits the *contrast resolution*. Signals of similar strength are binned in the same digital representation. As a consequence, very subtle differences cannot be captured. The size of the individual image elements of the detector is another factor that limits the *spatial resolution*. In particular, this carries weight when the object is placed close to the detector. For higher magnifications the focal spot is the limiting factor. Even though the crystals of the scintillating layer are applied to direct any photons directly to the underlying photo diode it is possible that some of them escape and hit other image elements of the detector. This in-detector scatter is another source of artifacts which harm the effective spatial resolution.

There are other types of detectors such as photon counting detectors. Here, every image element emits a signal when hit by a single photon [38]. These detectors promise a higher possible spatial resolution due to smaller image element sizes, a higher contrast resolution due to a better conversion efficiency, reduced beam hardening and metal artifacts due to an equal weighting of all photons (of different energies), and as they do not require a scintillating layer they do not suffer from afterglow. Yet, there are still some challenging aspects to solve. For example, these detector elements have a hard time dealing with the number of photons that arrive at the detector each second. Furthermore, there is an increase in in-detector scattering due to effects like partial energy deposition and Compton scattering which can cause the same photon to be counted twice—in different image elements [39]. Throughout this work we only consider flat panel detectors because of their prevalence in the industrial market.

The Exposure. Now we almost have all the ingredients necessary to simulate a realistic radiograph. However, the projections would be too perfect as we are missing the condiment, i. e. the noise which occurs during an actual exposure. The emitting of X-rays at the source, the attenuation in the object, and the sensing of the remaining intensity at the detector are all stochastic processes and, hence, subject to noise: In the source, within the focal spot, there is a large number of atoms of the target material each of which can be hit by an accelerated electron (and in consequence emit a photon) at a very low probability. Therefore, the photon emission of the source can be described by a binomial process. The *Poisson limit theorem*, in consequence, tells us that the generation of X-rays can be approximated by a *Poisson distribution*. Due to the quantum nature of the photons and because we can consider the object as a sequence of disjoint sections, where each section has uniform absorption properties, the different attenuation processes in the object can be statistically described as a binomial process, too. Finally, in the detector, we have similar processes: each of the atoms in the scintillating layer can be hit by a photon (and as a result emit visible light which is then detected by the photo diode) at a very low probability. Again, this describes a binomial process. The binomial selection drawn from a Poisson distribution follows a Poisson distribution itself. Therefore, we can describe the overall process of exposure with a Poisson distribution. For a precise derivation, we refer to [33]. This means for the simulation of a projection, we have to compute the theoretical intensity n that would be measured at the detector but draw the actual gray-value according to a Poisson distribution with $\mu = n$.

A measure of the quality of a projection is its signal-to-noise ratio (SNR), where $SNR = \frac{\mu}{\sigma}$. The characteristic property of our Poisson distribution is $\mu = \sigma^2$. With $\mu = n$, the expected number of photons which is a measure for the intensity of the X-ray radiation, we obtain $SNR = \frac{n}{\sqrt{n}} = \sqrt{n}$. This means that—in the best case—the quality of the image is improved only by the square root of the intensity of the X-ray radiation, i. e. to double the SNR we have to use four times the dose.

Additionally, the projections suffer from other sources of noise, which, for instance, emerge from the read-out electronics of the detector. This noise can be modeled by constant *Gaussian noise* [40]. During simulation the normally distributed noise is superimposed on the final projection.

But before we dive deeper into the different artifact types which impede an automated inspection, we need to reconstruct the volumetric information of the CT scan from the individual projections.

2.1.2. CT Reconstruction

In the previous section we described the basic setup to obtain the information about the attenuation of an object at different angles, which is necessary for CT. Now, we need to reconstruct the three-dimensional volume of the CT scan from the individual two-dimensional projections.

In other words, we would like to compute a volumetric image of our object, which explains all the projections we captured. In mathematical terms we, therefore, have to solve an inverse problem, i. e. drawing a conclusion about the cause from an observation. Because these problems are hard to solve—if solvable at all—we either have to invest plenty of computation time or make some relaxations. A widely used algorithm for the three-dimensional reconstruction is the Feldkamp algorithm (or FDK, short for Feldkamp, Davis, and Kress), a *filtered back-projection (FBP)* algorithm, which projects the captured intensities from the detector back into the object space (see Equation (2.2)). Its principal idea is to approximate the reconstruction of the three-dimensional object by treating

the three-dimensional cone-beam reconstruction as a set of two-dimensional fan-beam reconstructions, one for each detector row. This leads to a blurring outside the trajectory plane, which becomes stronger the bigger the angle between the detector row and the trajectory plane gets (see Figure 2.4a and Section 2.1.3). The approximation, however, allows for a fast and parallelized computation, which contributed to the widespread use of this algorithm.

$$f(x,y,z) = \underbrace{\int_0^{2\pi} \frac{D^2}{(D + x\cos\beta + y\sin\beta)^2} \underbrace{\underbrace{\left(\frac{D}{\sqrt{D^2 + \xi^2 + \gamma^2}} p_\beta(\gamma,\xi) \right)}_{\text{pre-weighting}} * h(\xi)}_{\text{filtering}} \, d\beta}_{\text{weighted back-projection}} \tag{2.2}$$

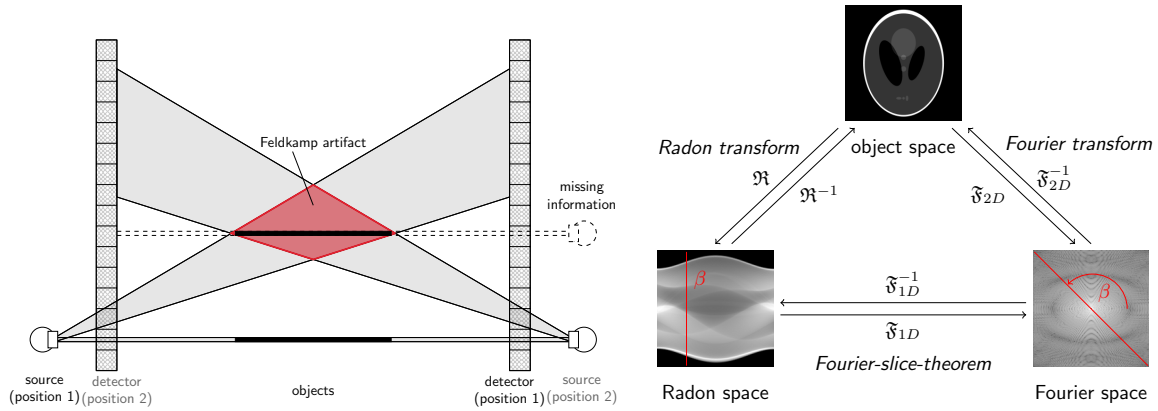
Note that we only consider the case of having a flat-panel detector as it is more common to industrial CT. The curved detector panels which are often used in medical devices require an adapted algorithm. The Feldkamp algorithm can be divided into the following three steps:

1. *Weighting*. The individual pixels in the projections are circularly weighted according to their fan-angle (which is the opening angle defined by the pixel ξ and the center of the detector row), their cone-angle (which is the opening angle of the detector row γ to the center row, i. e. the trajectory plane), and the distance D from the focal spot to the detector in order to cope for variations in the intensity as each ray has a different penetration length through the field of measurement. Note that if we do a bright image correction, we do not need this weighting step, because we obtain more detailed weights from the bright image, which further consider the properties of individual detector pixels.

2. *Filtering*. Each detector row is filtered with a one-dimensional high-pass filter kernel h to cope for the over-emphasis of low frequencies, which is explained by the Fourier-slice theorem (see down below).

3. *Back-projecting*. The final step is to compute the actual attenuation value f for each voxel denoted by its coordinates x, y, and z by summing over all projections p_β, which were taken at the rotation angle β. Note that f can be sampled at arbitrary positions, but it is highly recommended that the voxels should not be too small. The ideal voxel size is given by the pixel size of the detector and the magnification of the scan geometry. For smaller voxels the available information is only interpolated. Using bigger voxels, in contrast, allows to bin more information into a single voxel, which leads to a better contrast resolution. The back-projection is further distance weighted due to the divergence of the rays of the fan-beam. The rays of a fan-beam pass different voxels at different depths in different angles. Therefore, they contribute differently to each voxel and need to be weighted accordingly.

There exist plenty of variations of this three-dimensional FBP algorithm for the reconstruction of cone-beam CT, which mostly differ in small implementation details. Most of them utilize accelerating hardware like a graphics processing unit (GPU) to massively parallelize the reconstruction: The pre-processing can be parallelized over the projections, the back-projection over the z-slices.



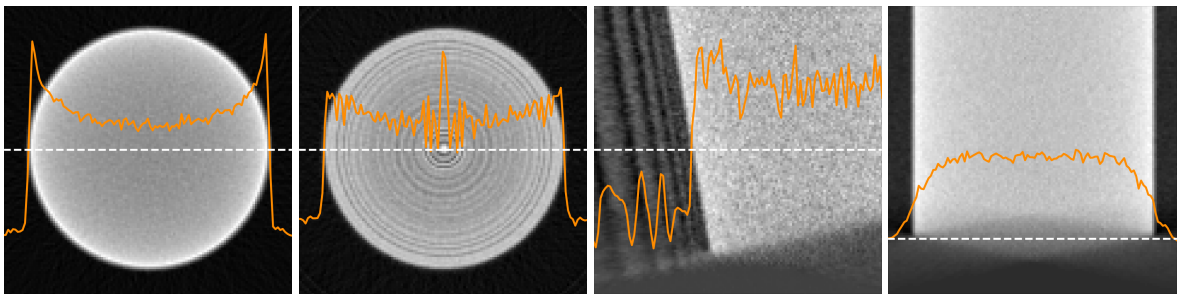
(a) (b)

Figure 2.4.: (a) Feldkamp artifacts (red area) occur in the reconstructed CT scan as a simple circular trajectory does not sample the Radon space completely. The missing information can be recovered by additional projections outside the trajectory plane, for example, by moving the source upwards (as indicated by the dashed lines) or adding a second circular trajectory which is orthogonal to the first one. (b) The Fourier slice theorem forms the connection of the Radon space and the object space in the Fourier domain, this figure illustrates the two-dimensional case as an example. The one-dimensional Fourier transform $\mathfrak{F}_{1D}$ of a projection taken at the angular step β corresponds to a slice through the two-dimensional Fourier transform $\mathfrak{F}_{2D}$ under the angle β. With all projections going through the center of the Fourier space, this explains the over-emphasis of low frequencies in the center of the Fourier transformed object.

Throughout this work we use the FBP implemented in the CT reconstruction module of VGSTUDIO MAX (Volume Graphics GmbH).

Radon Transform. The Radon transform is the foundation of CT, telling us that any two-dimensional function $f(x, y)$ (in our case f describes a plane in our object) can be reconstructed from the mean of all one-dimensional line integrals $p_\beta(\xi)$ (in our case p_β describes the projection at the angular step β). The principal goal in CT is to obtain f from a set of p by inverting the Radon transform. In the three-dimensional case we need to consider that a single circular trajectory does not fully sample the complete Radon space, which is particularly noticeable for large objects, which require wide opening cone-angles: In regions which are out of the trajectory plane we do not get a parallel view through the object and, thus, cannot precisely determine whether we are dealing with a flat surface or a conical structure (see Figure 2.4a). This means that the reconstruction can only be an approximation.

Fourier-Slice Theorem. The Fourier-slice theorem connects the Radon space spanned by p and the object space f within the Fourier domain (see Figure 2.4b). The one-dimensional Fourier transformed projection p_β taken at the angular step β is the slice with angle β in the two-dimensional Fourier transform F of f. In consequence of taking the projections from many angles the sampling within the low frequencies (in the center of F) is much denser than the sampling within the high frequencies. This explains the over-emphasis of low frequencies and motivates the high-pass filtering of the projections before back-projecting them.



(a) (b) (c) (d)

Figure 2.5.: Different types of artifacts: (a) cupping artifacts in an actually homogeneous material caused by beam hardening, (b) ring artifacts caused by an insufficiently calibrated detector, (c) aliasing artifacts caused by radial under-sampling using too few projections, and (d) Feldkamp artifacts caused by the simplifications introduced by the reconstruction algorithm. The orange line shows the gray-value profile which is sampled at the dashed white line.

Shepp-Logan Filter. To suppress the over-emphasis of low frequencies, the projection data p is filtered with a high-pass filter h, which is usually done in the Fourier domain. However, we do not use a simple ramp filter as it would introduce so called sidelobes, i. e. echos of the filter in actually homogeneous regions of the image. Furthermore, we need a band limitation of the filter to avoid an accidental over-emphasis of the noise in higher bands. The most commonly used filter, that satisfyingly fulfills all these demands, is the Shepp-Logan filter which (in the Fourier domain) combines a rectangle with a sinc-function, i. e. $H(q) = |q|\text{rect}(q)\text{sinc}(q)$.

Besides the FBP there are other algorithms like the iterative algebraic reconstruction, which yields less artifacts but is significantly more computationally expensive, or the helix CT, which acquires a full sampling of the Radon space and, therefore, allows for a direct reconstruction but requires a different CT setup.

2.1.3. Image Artifacts Impede The Detection

Due to the physical limitations it is hard to obtain a CT scan of high image quality. Usually, a CT scan contains a broad variety of image artifacts. These artifacts are particularly challenging when it comes to an automated inspection, as they introduce fluctuations in the gray-values which easily can be confused with actual features like defects. There is a long list of artifacts that occur at different strengths in any CT scan. The ones which influence the automated detection of defects the most are listed in this section.

Noise Wild spontaneous fluctuations occurring in the gray-values of a volume are probably caused by noise. As mentioned above there is plenty of source for noise, which all depends more or less on the number of photons arriving at the detector. Therefore, it is important to obtain a good illumination of the detector. Especially in the dark regions of a projection, i. e. where the material attenuates most of the radiation, we need to ensure that enough photons arrive at the detector and stay above a certain threshold which should be about 5 % to 15 % of the un-attenuated radiation. Furthermore, it is recommendable that the un-attenuated intensity is at about 90 % of the available detector range [37]. The portion of coherent noise, which is,

for example, introduced by the electronics in the detector, can be explained away by longer exposure times, i. e. taking more photons into account. If there is still too much noise in the projection images, the averaging of multiple projections helps to get rid of the coherent noise produced by the detector. This is especially important as we cannot use arbitrarily long exposure times without introducing other artifacts (e. g. focal drift, see below). The incoherent noise which occurs due to the stochastic processes during the attenuation (e. g. due to scattering) and during the detection (e. g. during the conversion to visible light), in contrast, cannot be mitigated by averaging multiple projections.

Cupping If the gray-values in a CT scan of an actually homogeneous material appear to be brighter in the outer regions of the CT scan and become darker towards the center, it is called a cupping artifact (see Figure 2.5a). These artifacts are usually caused by *beam hardening*. When passing through the material of the object under examination, the outer regions receive the full polychromatic spectrum of photons. As mentioned above, photons of lower energies are preferably attenuated. Therefore, the inner regions only receive photons of higher energies, i. e. the hard part of the polychromatic spectrum, which cannot be attenuated as easily. Hence, the total attenuation on the inner parts is lower, the gray-values in the reconstructed CT scan become darker. Filtering the spectrum at the source helps to mitigate the effects of beam hardening by suppressing low energetic photons *ab initio* so that all regions of the object under examination receive only the hard part of the polychromatic spectrum. Figure 2.3a shows the comparison of an unfiltered and a filtered spectrum of a tungsten target at the same voltage and current. However, this comes at the cost of an overall reduced number of photons.

Rings The radiation is focused on the center of the detector which means that its edges are less illuminated. Moreover, due to differences and static noise in the electronics the detector also yields a signal even if the source is turned off. To obtain a meaningful reconstruction we need to compensate the decrease in illumination and the unwanted signals by calibrating the detector. The decrease in illumination towards the edges of the detector is covered by the bright image I_B which is acquired by averaging several projections without any object present. The unwanted signals are covered by the dark image I_D, i. e. a projection taken without any object present and with the source turned off. The corrected projection image I, which is used for reconstruction, is obtained from the raw projection image I_{raw} as defined in Equation (2.3). If this calibration is done poorly, for example, by acquiring a noisy bright image, the same error is introduced to every projection. These errors manifest in the reconstructed CT scans as concentric rings of varying gray-values, i. e. the ring artifacts (see Figure 2.5b).

$$I = \frac{I_{\text{raw}} - I_D}{I_B - I_D} \tag{2.3}$$

Another source of ring artifacts are dead detector pixels, which always yield the same response independently of their exposure. (Usually, they always yield no signal, i. e. zero, or the maximum possible value.) The dead pixels need to be eliminated for reconstruction as well, e. g. by interpolating between unimpaired neighboring pixels.

Scatter Scatter causes similar artifacts in the reconstructed CT scan as beam hardening. As scatter alters the direction (and energy) of photons, it causes some detector elements to receive less intensity than expected while especially those detector elements which are behind highly

attenuating material receive more intensity. Beside the scattering within the object there are also back-scattered photons from the environment, e. g. the radiation protection chamber or other components of the CT system, and there is in-detector scattering. Scatter is a significant source of image artifacts in CT scans and is hard to mitigate. The amount of scattering from the object and the environment can be reduced by using a proper collimation which absorbs photons that do not originate directly from the source. Nevertheless, there still remains the in-detector scatter which additionally blurs the edges of the projected object and, thus, is not negligible [41].

Blurring Depending on the magnification, either the size of the *focal spot* or the size of the *detector elements* has the most significant influence on the (un-)sharpness of the CT scan. The larger the magnification, the more significance has the focal spot size ($d_{\text{focal spot}} \cdot (m-1)/m$). The closer the object is to the detector, the more significance has the pixel size of the detector (d_{pixel}/m). d_x refers to the diameter of the focal spot and the diameter of a detector pixel, respectively. On top of this, there are many other factors that contribute to the blurring of a CT scan, such as subtle movements of the object or the components of the CT system cause a deviation from the setup which is expected for reconstruction and, therefore, cause additional blurring in the final CT scan. A *geometrical imprecision* can be caused by a tilted rotation axis, deviations in the rotation steps, or simply micro movements of source, object, or detector due to vibrations induced by the environment or the system itself. Another source can be *focal drift*: During long-time exposures the focal spot can start to wander or change its appearance. This contributes to further blurring and cannot be corrected. Instead of increasing the exposure time of a single projection, it can be beneficial to average multiple projections from the same angle to increase the effective exposure time while avoiding a focal drift. Furthermore, image artifacts like scattering contribute to the blurring of a projection as mentioned above.

Aliasing The acquisition of a CT scan is a sampling process and, thus, we have to fulfill the sampling theorem. This means that if the number of projections falls below a certain minimum number, the object is radially under-sampled and we end up with aliasing artifacts, which manifest as shadows of the object in the CT scan (see Figure 2.5c). The minimum number of projections corresponds to the minimum number of detector elements which are necessary to capture the object under examination. This is explained by Equation (2.4), where N_p is the number of projections, D_{min} the minimum number of detector elements, d_{FOM} is the diameter of the field of measurement, and d_{detail} the diameter of the detail we would like to resolve. However, in practice the number of projections required to obtain an acceptable reconstruction depends on plenty of other parameters like the opening angle of the source or the shape of the detail we would like to resolve.

$$N_p \approx D_{\text{min}} > \frac{2 d_{\text{FOM}}}{d_{\text{detail}}} \tag{2.4}$$

Partial volume If the object under examination only partially covers a voxel, during reconstruction this voxel partially receives information from the un-attenuated radiation and the attenuated radiation. If the object exactly covers half of the voxel, the FBP reconstruction algorithm would expect the signal to be half the signal of a fully attenuating voxel, i. e. we compute $\ln(\text{avg}(I_0/I))$ where it would actually be $\text{avg}(\ln(I_0/I))$. This leads to an underestimation of the

attenuation and, therefore, to artifacts which can appear as streaks in the CT scan if the partial information comes from all projections.

Feldkamp As described in Section 2.1.2 the FDK reconstruction algorithm is only an approximation and becomes less precise when moving farther away from the trajectory plane (compare Figure 2.4a). Figure 2.5d shows an example of the resulting artifacts in the reconstructed CT scan, the so called Feldkamp artifacts. To avoid these artifacts we need to switch the reconstruction method and the acquisition of the projections, for example, by turning to a helical scan pattern.

Having described the principal CT setup and the reconstruction process as well as knowing about the cause of image artifacts, we can derive some guidelines for making a good CT scan. This will further help us to determine a good baseline for our simulations. In general it is better to increase the voltage, in order to obtain a good penetration in all regions of the object so that we get at least some contrast in all occurring thicknesses of the object. Moreover, higher voltages (in combination with a suitable pre-filter) will mitigate the effects of beam hardening. The limited contrast steps will be mitigated by the total number of projections used to reconstruct the CT scan. Hence, the more projections we make the better will be the CT scan. Furthermore, this avoids running into aliasing artifacts due to under-sampling. To make the most out of the scarce information of the few photons arriving at the detector, it is better to use less pixels and so bin more photons in a single detector element to receive a stable signal without a high noise induction. The proper detection of defects usually does not require a detector with more than 2000 by 2000 pixels. Finally, the longer the total exposure time the better will be the final CT scan.

2.1.4. Towards a Full In-line Inspection

When it comes to a full inspection of an entire production line, the tight cycle times additionally limit the time we have for the acquisition and inspection of a CT scan. Here, we usually have to cut back on data quality. Coming from a high quality scan that follows the guidelines described above, we can reduce the exposure time to achieve the desired high throughput, having only some minutes for a CT scan. In exchange, we have to decide between keeping a high spatial resolution or keeping a high contrast resolution. We either have to introduce artifacts to the CT scan (most notably noise), or we have to accept more blurring and fewer voxels. Partially, the adverse effects of reducing the scan time can be mitigated by using more sophisticated (and more expensive) components, the laws of physics, however, cannot be overridden. As Figure 2.6 indicates, a CT scan is always a consideration of spatial resolution, contrast resolution, the cycle time, and (up to a certain extent) the facility costs [42].

As mentioned in the beginning, despite all the efforts, inspecting each part before it leaves the production site can be beneficial not only for safety relevant parts. Ruling out defective parts in early stages of the production process saves time and money. The tough cycle times, however, render a manual inspection infeasible. It requires automated inspection techniques which are capable of dealing with the low achievable data quality, supporting the quality specialists in their assessment. In the following section we explore the state-of-the-art methods in automated defect detection.

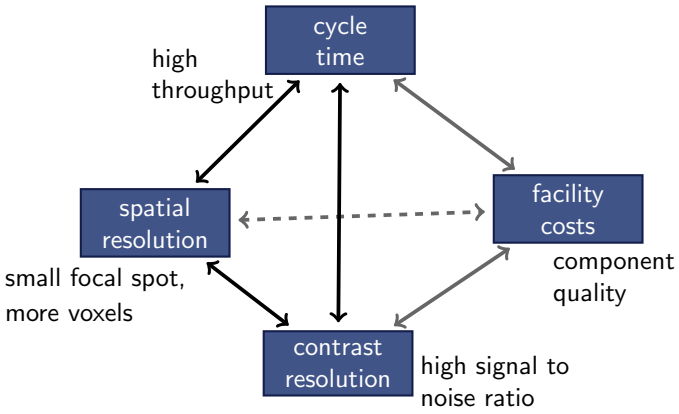


Figure 2.6.: The X-ray testing pyramid: Acquiring a CT scan is always a consideration of spatial resolution, contrast resolution, and desired cycle time. Up to a certain extent the quality of a CT scan can be influenced by better components, the laws of physics, however, cannot be overridden [42].

2.2. Automated Defect Detection

In CT scans defects manifest in a variation of the gray-values within the material region. These variations can be very subtle and, thus, have the potential to vanish in the noisiness of the data. A manual inspection which involves a human inspector slicing through the CT scan layer-by-layer looking for these variations takes time. An inspection can easily fill a complete working day keeping an expert busy—especially because we need to plan with pauses. An expert cannot stare eight hours nonstop at the data. Furthermore, the manual inspection requires special hardware like standardized monitors which precisely display gray-values always the same way. Therefore, we need algorithms which support the experts in their work and automatize the inspection process in order to enable a full in-line inspection. The state-of-the-art algorithms usually go for an *anomaly detection* approach: They identify the material region and look for deviations in the gray-values. While presenting the most prevalent approaches in automated defect recognition, we also collect requirements for an improved method, which we will consider during the development of our deep learning defect detection method.

Requirement: The first requirement we can add to our list is the inference speed. The model needs to be very fast to keep up with the short cycle times of an in-line inspection.

2.2.1. Image Processing Methods

The typical approaches to defect detection are threshold methods which yield a response if the deviations in the gray-values exceed a certain threshold. The high artifact-affliction of CT scans usually does not allow us to use a global threshold for the entire CT scan. Hence, the threshold is defined on a relative scale and *locally adaptive* methods which consider the gray-value distribution within the neighborhood of each voxel are used to segment the defects. The results are usually limited to the material region of the CT scan to eliminate potential false positive responses in the

background. This usually solves the task for CT scans of high data quality, which we encounter, for example, in quality laboratories when doing a sample inspection. In Section 4.1.3 we present a locally adaptive thresholding technique which we further use as one of our baselines to compare our deep learning algorithm with.

The next step is to do a connected component analysis to find the individual defect instances and compute some statistics like their volume, their surface area, or their sphericity. We can then filter the instances according to their statistics. Tiny instances, for instance, are more likely to be false positive responses due to image noise and can be ignored—especially if they are below a certain size which renders them uncritical anyway.

However, it still can happen that we obtain too many false positive responses or completely miss out critical defects in particularly artifact-afflicted regions of the CT scan. In such a case we can further subdivide the analysis region and use different sets of parameters for each region. We can imagine that tuning all the parameters of the (locally adaptive) threshold methods and the filters is a tedious and time-consuming job. Furthermore, this is hard to automate. We need to guarantee that each CT scan of our inspection system shows the same gray-value distribution. If we slightly deviate from the optimal operating point, our inspection method is likely to produce false results.

Requirement: We need a method which works independently of local variations in the gray-values and without the necessity to tune an excessive amount of parameters.

2.2.2. Reference-based Approaches

More sensitive and more precise are reference-based anomaly detection methods. Here, the CT scan of each inspected part is compared to a defect-free specimen, the so called "*golden part*" [43–45]. The golden part captures the same artifacts as the CT scans of the individual parts under examination. Hence, it is easier to distinguish artifacts from actual defects. This procedure requires all CT scans to capture the part under examination at the exact same angle and with the exact same parameters to avoid false alarms. Therefore, this approach is only used for the in-line inspection of serial parts utilizing robots to ensure an exact placement [44]. Another benefit of this approach is that it is not limited to defects in the material region but also can detect other deficiencies like cold runs. The golden part is, for example, computed as the mean or median of multiple (> 100) CT scans of the same part [43] or a simulated replica of the real part [44].

The method described in [45] is a more general approach which does not require a precise repeatable scan process but a higher scan quality that allows for a precise alignment. Their pipeline comprises five steps: (1) the enhancement of the CT scan, (2) the alignment with the golden part, (3) the computation of the difference image, (4) the derivation of the segmentation mask, and (5) the refinement of the segmentation mask. In step (1) noise in the CT scan is reduced using a Wiener filter (which suppresses noise by means of the mean squared error) and the gray-value distribution is adjusted by performing a histogram equalization. The alignment step (2) compensates for deviations in the orientation of the scanned part based on the gray-value distribution of the CT scan and the golden part. An optimization algorithm estimates the parameters of an affine transformation reducing the cost function given by the divergence of the two gray-value distributions. Then, the element-wise

absolute difference of the CT scan and the golden part is computed (step (3)). The segmentation mask results from thresholding the difference image (step 4). In [45] the threshold is chosen to maximize the sum of the entropy values of foreground and background. As the segmentation mask might still contain some false positive alarms induced by image noise, in step (5) a morphological opening filters smaller indications. Finally, the remaining anomalies can further be classified whether they are pores, cold runs or other defects.

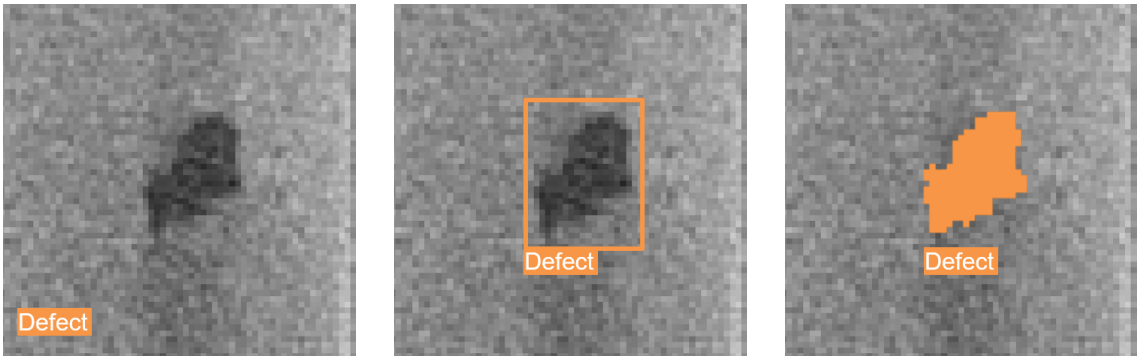
Besides the tedious setup of the inspection pipeline, this method has the drawback of being heavily tailored towards a specific part and casting geometry. The smallest change in the casting process or the smallest deviation in the inspection process can raise a false alarm. The casting process, however, is changed by the casting engineers on a regular basis: For example, they often have to change the gates or risers, which are only removed in post-processing steps after the inspection, to compensate for environmental changes or minor changes in the material composition. With that the new CT scans heavily deviate from the golden part in the region of the intentional changes, which requires a time-consuming update of the golden part.

Requirement: We need a method which does not depend on a reference part and, therefore, is invariant to (minor) changes in the part under examination.

2.2.3. What About Machine Learning?

We observed that only in recent years machine learning algorithms were introduced to directly solve NDT-related x-ray CT tasks, e. g. in [46]. Most machine learning algorithms in NDT literature focus on processing two-dimensional radiographs like [47] or tackle the question whether a part (or a pre-selected area of a part) is “okay” or “not okay” like [48]. In terms of the detection and segmentation of casting defects in three-dimensional CT scans we found an approach that utilizes a traditional method to find defect candidates in the CT scans and then uses a random forest to reduce false positive responses [49]. We explain this method in more detail in Section 4.2.1. One reason for the absence of machine learning approaches for the detection and segmentation of defects could be the lack of a suitable labeled data set which we address in this work by using simulations.

Most other approaches use machine learning as a pre-processing step, trying to improve the data quality of the CT scans to pave the way for simpler detection methods. Removing noise artifacts from CT scans is of interest not only in the industrial sector. In medical imaging, the ability of removing noise on the software side allows for lower doses. Therefore, the denoising of low dose CT is a highly explored field of application [50, 51]. Yet, these image enhancement algorithms still include the risk of removing small but important details from the image, as they try to optimize the overall image quality in terms of a global cost function. Despite most of them being edge preserving, small defects in homogeneous material regions can be removed erroneously. Nevertheless, these enhanced CT scans can be used for a huge variety of tasks like measurement and nominal/actual comparisons and ease the processing of the data by a human expert. Other artifact reduction techniques like the deep scatter estimation [52] work on the projection data instead of the reconstructed CT scan. For example, the deep scatter estimation uses a deep neural network to approximate the result of a Monte-Carlo scatter estimation from a two-dimensional projection, which is then used to remove the scatter component from the projection. This approach is more likely to also preserve small details



(a) classification (b) localization (c) segmentation

Figure 2.7.: These images illustrate the differences between image classification, object localization, and (semantic) segmentation. In (a) we only distinguish between defective or not for the CT scan as a whole, in (b) we locate the defect within a given CT scan, and in (c) we precisely determine the boundaries of the defect which enables us to further compute derived metrics like its volume or sphericity. Note that from left to right not only the amount of information increases but also the necessary labeling effort.

as it does not alter the data directly but only removes the portion of information in the projections which contributes to scatter artifacts.

Requirement: We need a method which is capable of dealing with image artifacts—particularly with the noise present in fast CT scans.

2.3. Semantic Segmentation

Semantic segmentation is the combination of localization and classification on a pixel (or voxel) level. A *segmentation* describes a set of individual image points belonging together according to a given criterion. This can be the clustering by local proximity and similarity of properties, for instance, a similar gray-value or color intensity distribution, i. e. a super-pixel segmentation [53, 54], or the detection of edges which separate different regions in an image [55]. While both approaches segment areas in an image which are inherently similar, theses segmented objects do not necessarily need to represent semantic objects. For example, such an algorithm could split a person into upper body (shirt) and lower body (trousers), i. e. an over-segmentation. Therefore, we need to assign one of a limited number of class labels to each pixel, giving the pixels a *semantic* meaning, but we do not need to distinguish between different instances (which would describe an instance segmentation). Figure 2.7 illustrates the differences between classification, localization, and (semantic) segmentation. Note that the increasing richness in details of the labels also necessarily entails more annotation effort.

Semantic segmentation is a typical machine learning task in computer vision. Using this scenario as an example, this section provides a short introduction to machine learning and briefly introduces the tools which are necessary throughout this work. But let us discuss some basic terminology first.

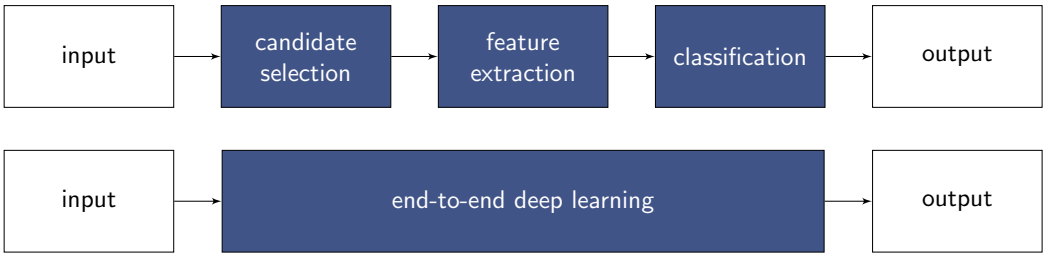


Figure 2.8.: When moving from traditional machine learning pipelines to end-to-end trainable deep learning, the major effort of creating a successful method is shifted from the sophisticated design of proper features and candidate selection algorithms towards the creation of a meaningful training set and the proper regularization of the powerful deep model.

While segmentation itself can be an *unsupervised* process (using, for example, a clustering method), semantic segmentation requires a set of labeled data from which the method can derive the necessary values for its parameters—it is a *supervised* learning process. Therefore, common to all supervised methods is the need for labeled training data. Besides the labeling effort mentioned above, this comes with certain risks: What is not covered by the *training set* is rather unlikely to be recognized by the trained model. That means there is the risk to introduce an unwanted *bias* to the model by an inconsiderate data selection. Furthermore, if the training set is too small or too undifferentiated, there is the chance that the model is tied too closely to the data used for training, such that it completely fails on unseen data, i. e. the model *overfits* to the training set. To check for overfitting, the total amount of labeled data is usually split in a training set and a (smaller) *validation set*. If the trained model performs equally well on the training set and the validation set, which was not considered during training, it indicates an ideal fit. The model is more likely to perform well in practical use. One way to prevent a model from overfitting is to introduce additional boundary conditions and penalty parameters during training. This process is referred to as *regularization*. Too much regularization, on the other hand, can cause poor performance as well, i. e. *underfitting*.

2.3.1. Fully Convolutional Neural Networks

In the last decade the trend in computer vision—and more generally in machine learning—moved from the traditional machine learning pipelines towards *end-to-end trainable* deep learning methods. While the traditional approaches require a lot of domain knowledge to find the best algorithms and methods for the selection of candidates and the extraction of sophisticated, hand-designed features, in *deep learning* approaches the effort is shifted towards the creation of training data and the proper regularization of the powerful models which comprise candidate selection, feature extraction, and classification (see Figure 2.8). In the traditional methods a “shallow” classifier takes a decision based on few explainable features which are extracted from the input data. In contrast, deep end-to-end trainable methods are trained on a vast amount of annotated training data to solve the task by learning how to extract features and how to draw conclusions based on these features at the same time. Thereby, “deep” refers to the huge number of trainable parameters or the large number of consecutive layers they are arranged in. The successive processing of the input by a large number of layers not only increases the capacity of the model by adding further parameters, it also enables the model to build an abstraction of the input step-by-step.

While neural networks—at least in principal—were known to the machine learning world since the 1960s [56], it took three major steps for deep learning to become successful, beginning with LeNet [57] for recognizing handwritten digits and AlexNet [58] for image classification in the late 1990s and early 2010s, respectively:

Huge labeled data sets As mentioned above, deep learning models have a huge number of free parameters which need to be trained. The training process, therefore, requires lots and lots of labeled training data—the deeper the model, the more training data is necessary. Thus, the emergence of large labeled training sets for image classification, like ImageNet [59] or Microsoft COCO [60], and scene understanding, like PascalVOC [61] or KITTI [62], significantly contributed to the success story of deep learning models.

Powerful acceleration hardware Another important aspect that comes with the increased amount of parameters is the number of operations that have to be computed in order to train the models and to obtain a result during inference. Fortunately, deep learning models mostly comprise linear algebra and element-wise operations. These operations allow for massive parallelization. Hence, they are well suited for modern GPU architectures, which are particularly designed to perform the same operation on a large amount of data in parallel [63]. With the rise of open source frameworks it was now possible to conduct experiments on standard consumer hardware—without the necessity to have access to an expensive compute cluster.

Efficient training algorithm The final ingredient is the development of an efficient training algorithm which is necessary to tune the millions of free parameters of a deep learning model. The main idea of this algorithm is to propagate the error which we observe at the output layer backwards through the network to compute the error gradient for each parameter on the way, i. e. the back-propagation algorithm, and then use a stochastic gradient descent (SGD) algorithm to actually update these parameters [64]. More on the training process below.

Building a Deep Learning Model. A deep learning model arranges a vast amount of simple linear classifiers, the neurons, in a large number of consecutive layers. The *artificial neuron* follows its biological model albeit in a very simplified way: all input values are combined by a weighted sum (plus adding a bias term) and then passed through a non-linear function, the so called activation function, which decides about the output of the neuron. This output is then passed on to the neurons in the next layer. In contrast to real neurons we are not dealing with a continuous process but with discrete time steps.

A typical *activation function* is the rectified linear unit (ReLU) [65]. It convinces by its simplicity: It is zero for negative input values and passes on the input value otherwise. Compared to other activation functions like the hyperbolic tangent it does not suffer from a vanishing gradient—except for negative input values, which, however, can be beneficial to enforce sparsity. If "dying" neurons need to be avoided, there exist plenty of variations of the ReLU, for example, the leaky ReLU which outputs the input value reduced by a given factor $c \neq 1$ for negative input values, where c is usually smaller than 1 [66].

The weights (and the bias terms) of the neurons are the trainable parameters of the deep learning model—the artificial neural network. The neurons (or nodes of the network) are connected in a series of consecutive layers. The weights can be seen as attached to the edges of the resulting graph.

Here, we can choose from different types of layers in which the individual neurons are arranged differently. The most important types are described below.

Fully connected The fully connected layer is a set of nodes each of which is connected to every node of the previous layer. This type is common to classification networks: in the output layer of the network, each node represents a class and is trained to yield large output values, if the input corresponds to that class. The huge drawback is that these layers are tied to a fixed input size, which introduces limitations to their applicability in computer vision tasks. Yet, it is possible to convert a fully connected layer to a convolutional layer using a kernel size of 1 (see next item). This, for example, allows to coarsely locate objects in larger images.

Convolution Especially in computer vision tasks the size of the inputs quickly gets out of hand. Therefore, the nodes are only connected to a small window of the input, which is then sled across the entire input. In every step, the nodes use the same weights to determine their output—they are *shared* over the input. This describes a simple convolution (see Figure 2.9a). The only difference is that, instead of using a predefined kernel, the weights of the kernel are learned to fit the data. The benefits are that we have to learn less parameters and that we are independent of the size of our input. The number of nodes again corresponds to the number of features computed. They are also referred to as *channels*. As the nodes only see a small portion of the input at a time, they are able to recognize certain features at any location of the input, which makes the features of convolutional layers *translation invariant*.

Depending on the kernel size k the kernel needs a certain amount of input values to compute an output value. Thus, the size of the input will be reduced by $\lfloor k/2 \rfloor$. If this reduction is not desired, the input values need to be *padded* by $p = \lfloor k/2 \rfloor$ (see Figure 2.9a). Usually, a simple padding with zeros is used to preserve the input size.

k further determines the *receptive field* of a convolutional layer, i. e. the part of the input that can be seen at once. With the concatenation of multiple convolutional layers the total receptive field of the neural network grows. For example, a model of two successive convolutional layers with $k = 3$ has a theoretical receptive field of 5. However, due to small weights and the selection process in pooling layers, which theoretically would at least double the receptive field (see next item), the effective receptive field is much smaller than the theoretical receptive field.

Pooling Pooling layers, which are hardwired layers to combine a certain amount of input values to a single output value, serve the purpose of *context aggregation* and lower the spatial resolution. This creates room for more features. In a pooling layer typically a window with size $k = 2$ is sled across the input with a stride $s = k = 2$, i. e. only considering every second position to reduce the spatial extent by half. The output value is either the maximum value (*max pooling*) or the average value (*average pooling*) of the input values.

Dropout Often listed as a separate layer, the dropout basically discards the results of the preceding layer element-by-element with a given probability p [67]. During training p is set to a value greater than zero (usually $p = 0.5$) so that the model only has a part of the information available. This acts as regularizer to avoid overfitting. At inference time p is set to zero, i. e. the dropout layer is skipped so that the model can operate on the full set of information to make a reliable decision.

Strided convolution Another approach to reduce the spatial resolution is introducing a stride to the convolutional layer, only considering every n-th position in the input (see Figure 2.9b) [68]. Here, the reduction method is not hard-wired. Instead, we let the model learn the necessary weights to figure out how to optimally combine the input values. The benefit, however, is also the drawback: we introduce more parameters which have to be trained.

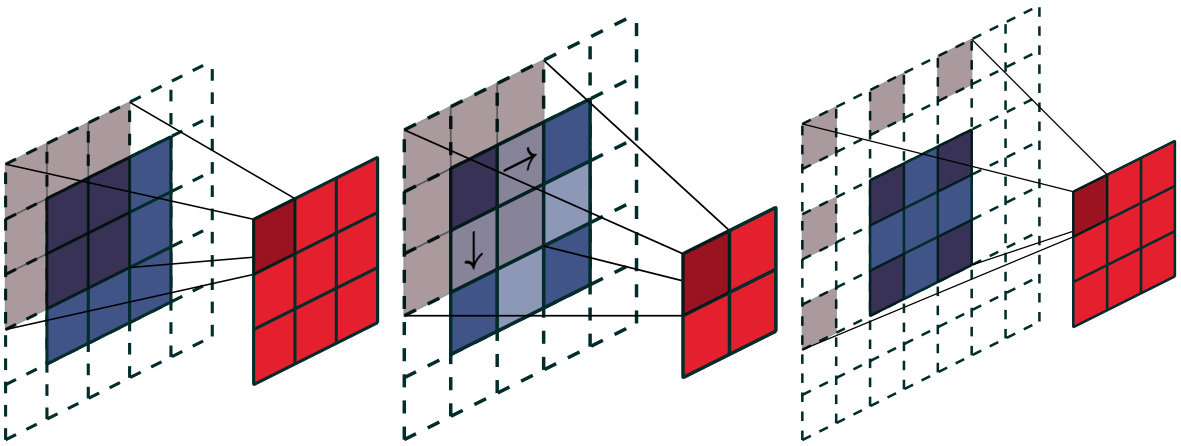
Un-pooling For a precise semantic segmentation the size of the output of our deep learning model needs to correspond to the size of the input. Thus, we have to increase the spatial resolution when done with the feature extraction, i. e. after the so called *bottleneck*. A straight forward approach is to simply multiply the values of the input. That means making four output pixels out of one input pixel or correspondingly eight output voxels out of one input voxel.

Transposed convolution This type of layer turns the tables: instead of combining multiple inputs to a single output, a single input is broadcast to multiple outputs. The individual broadcast results are combined by summing them up. This alone does not serve the purpose of increasing the spatial resolution. The next layer is only increased by $2 \cdot \lfloor k/2 \rfloor$ pixels. To be the equivalent to an un-pooling layer we again need to introduce a stride—this time in the output layer. As it happens that the broadcast output values only partially overlap, this method is prone to introduce *checkerboard artifacts* to the output. This type of layer is often referred to as convolution with fractional stride, because the stride on the input seems to be fractional to increase the output size, or erroneously as deconvolution, even though it is not a deconvolution in a mathematical sense.

Dilated convolution Instead of (or in addition to) reducing the spatial resolution for the aggregation of context information it is possible to spread open the input of the filter kernel (see Figure 2.9c). These dilated convolutions further widen the receptive field of the model and so enable the processing of more contextual information.

Besides the layer types described above, which form the most principal building blocks for the neural network architectures used for classification and segmentation tasks, there exist plenty of other types which are not discussed here. Furthermore, compositions of the above types are sometimes listed as individual layer types. One example for a compound layer is the *residual layer*, which adds the result of several consecutive convolutions (and non-linearities) as correction to its original input (compare Section 4.3.1 and Figure 4.14). The nesting of layers allows to comprehensively describe even more complex network architectures like the ResNet [69]. Additionally, *element-wise operations* (like an addition) and the concatenation of results are often listed as individual layer types.

By concatenating convolutional layers for the feature extraction, pooling layers for the context aggregation, and fully connected layers for the feature combination and classification, we obtain typical image classification networks like AlexNet [58] or VGG-Net [70]. In accordance with the name-giving layer type, the convolutional layer, this type of architecture is referred to as convolutional neural network (CNN). These were the first models to come close to a human-level performance on the ImageNet image classification challenge with 10 000 different classes [59]. Instead of using fully connected layers to obtain a single classification, we can up-scale the results with the help of un-pooling layers and transposed convolutional layers to obtain a classification per image element, i. e. a semantic segmentation. Prominent architectures are, for example, the U-Net [18] or the fully convolutional network (FCN) [12].



(a) standard convolution (b) strided convolution (c) dilated convolution

Figure 2.9.: Illustration of the different types of convolutional layers. The blue squares represent the input data. The white squares with the dashed outline are the padded input data. The red squares represent the output data of the convolution. The squares which are shaded in red are the values which the kernel uses to compute the single output value denoted in darker red. All three convolutions use a kernel size of 3×3. The strided convolution uses a stride of 2×2 (denoted by the light blue squares and the arrows) and the dilated convolution a dilation of 2×2.

The Training Process. In this section we briefly explain how the training process of a neural network works in principal, sticking closely to [71], to which we also refer for a more detailed mathematical derivation. To obtain a prediction we successively pass the input values on through the different layers of the network (the so called *forward pass*, outlined in Figure 2.10a). Of course, with an untrained model the results will be as random as the initialization of its weights. Hence, we need an optimization algorithm to train the weights to some values which match our task. The many non-linearities in the model, induced by the activation functions, render the optimization problem to be non-convex. This has the consequence that neural networks are trained using an iterative, gradient-based optimizer and, therefore, end up in a local optimum instead of finding the global minimum of the cost function [71]. However, as the neural network maps its input to a very high dimensional latent space any local minimum is approximately as good as the global optimum [71]. Here, we explain the typical optimization procedure for neural networks, called "training", along with some implementation details of specific optimizers.

As mentioned above, we use the *back propagation* algorithm to pass the information of the prediction error backward through the neural network. With that we can compute the error gradient for each weight and use a *stochastic gradient descent* algorithm to update the weights accordingly and expect the prediction error in the next forward pass to be smaller. The error of the prediction is determined according to a loss (or cost) function which is used to define the desired output of the neural network. The necessary loss functions will be explained in-place where necessary. For now it is only important that the loss function provides the initial gradient information. The back propagation algorithm now recursively applies the chain rule (of calculus) to trace this error all the way back to the input yielding a gradient for every trainable parameter in the neural network (as outlined in Figure 2.10b). With the gradient information $\mathbf{g}$ we update the parameters θ of our model such that $\theta_{t+1} = \theta_t - \alpha\, \mathbf{g}$, i. e. a gradient descent, where α is denoted as the learning rate. As mentioned before, neural networks

$\mathbf{h}^{(0)} \leftarrow \mathbf{x}$
for $k = 1, 2, ..., K$ **do**
  $\mathbf{a}^{(k)} \leftarrow \mathbf{h}^{(k-1)}\mathbf{W}^{(k)} + \mathbf{b}^{(k)}$
  $\mathbf{h}^{(k)} \leftarrow f\left(\mathbf{a}^{(k)}\right)$
end for
$\hat{\mathbf{y}} \leftarrow \mathbf{h}^{(K)}$

(a) forward pass

$\mathbf{g} \leftarrow \nabla_{\hat{\mathbf{y}}} \mathcal{L}\left(\hat{\mathbf{y}}, \mathbf{y}\right)$
for $k = K, K-1, ..., 1$ **do**
  $\mathbf{g} \leftarrow \mathbf{g} \odot f'\left(\mathbf{a}^{(k)}\right)$
  $\nabla_{\mathbf{b}^{(k)}} \mathcal{L} \leftarrow \mathbf{g} + \lambda \nabla_{\mathbf{b}^{(k)}} \mathcal{R}$
  $\nabla_{\mathbf{W}^{(k)}} \mathcal{L} \leftarrow \mathbf{g}\, \mathbf{h}^{(k-1)\mathsf{T}} + \lambda \nabla_{\mathbf{W}^{(k)}} \mathcal{R}$
  $\mathbf{g} \leftarrow \mathbf{W}^{(k)\mathsf{T}}\mathbf{g}$
end for

(b) back propagation

Figure 2.10.: During training we perform a forward pass (a) of the input data $\mathbf{x}$ through the K-layered neural network storing the activations $\mathbf{a}$ and the output $\mathbf{h}$ of each layer, where $\mathbf{a}$ is computed as the dot product of the output of the previous layer and the weights $\mathbf{W}$ of the current layer and adding the bias term $\mathbf{b}$ of the current layer. Then, $\mathbf{a}$ is passed through the non-linear activation function f to yield the output of the current layer. The output of the K-th layer is the final prediction $\hat{\mathbf{y}}$ of the neural network. A loss function $\mathcal{L}$ compares $\hat{\mathbf{y}}$ to the actual labels $\mathbf{y}$ of the input and its derivative with regard to $\hat{\mathbf{y}}$ gives us the initial error gradient $\mathbf{g}$ for the back propagation (b). For back propagation we iterate backwards through the K layers of the neural network, first updating $\mathbf{g}$ by the element-wise product of $\mathbf{g}$ with the results of the derivative of the activation function. Then, we can compute the gradient information for the parameters ($\mathbf{W}$ and $\mathbf{b}$) of this layers. Note that we also include a regularization loss $\mathcal{R}$ which is added (weighted by λ) to the gradient. The task of the regularization loss is to keep the weights in a proper value range which contributes to the avoidance of overfitting. Finally, we compute the derivative of the outputs of the next (or rather previous) layer with regard to $\mathcal{L}$ by computing the dot product of $\mathbf{g}$ with the transposed weights of the current layer. The gradient information for the parameters is stored for the update with the SGD algorithm. During inference, we only need the forward pass (a).

are trained with very large data sets. Training would take an eternity when computing the complete gradient information $\mathbf{g}$ for all data points in each iteration. Hence, we only use a small subset of the data points (a mini-batch) which is randomly chosen in each iteration and update the parameters of our model using an estimate $\hat{\mathbf{g}}$ of the actual gradient which is the mean gradient of the mini-batch. The larger the mini-batch we use in the (mini-batch) SGD algorithm the less noisy becomes the estimated gradient (even though this scales only sub-linearly) and the larger can be the learning rates.

A common problem is that the gradient estimated from one mini-batch can point in one direction, while the gradient estimated from the next mini-batch can point in another direction, sharing only a small portion pointing in the same direction. To address this, we can add a *momentum* information to the SGD algorithm. We introduce another variable $\mathbf{v}$ which corresponds to the "velocity" of the learning, i. e. the direction and speed of the parameter changes. In each iteration we update $\mathbf{v}_t = \beta_1 \mathbf{v}_{t-1} - \alpha\, \mathbf{g}$ and only with $\mathbf{v}_t$ we actually update the parameters $\theta_{t+1} = \theta_t + \mathbf{v}_t$. The parameter $\beta_1 \in [0, 1)$ determines how fast the learning responds to changes in the direction of the estimated gradient.

The second challenging aspect of training is the condition of the gradients: exploding and vanishing gradients can cause the training to fail. While here the topology of the loss surface and the depth of the neural network play an important role, the effects can be mitigated by choosing the right learning rate. Furthermore, the right learning rate can change during training. In the beginning, a high *learning rate* can be beneficial to quickly land near a local minimum. Then, already close to a low cost local minimum a low learning rate is necessary to actually find the minimum instead of meandering around it. Therefore, the learning rate is one of the most important, yet most difficult,

hyper-parameters to tune [71]. A simple decay of the learning rate over time is a good starting point to solve this issue. More sophisticated algorithms like *AdaGrad*, *RMSprop*, and *Adam* use *adaptive learning rates* for each parameter individually and, hence, allow for larger initial learning rates. These algorithms are basically some sort of evolution from one another, starting with AdaGrad, a coinage of "adaptive gradients", which weights the learning rate by the square root of the sum of the squared magnitude of all gradients ever occurred. The gradient history is updated by $\mathbf{r}_t = \mathbf{r}_{t-1} + \mathbf{g}^2$ and the parameters by $\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\mathbf{r}_t}} \odot \mathbf{g}$. In contrast, RMSprop (root mean square propagation) uses an exponentially weighted running average with an additionally controllable hyper-parameter β_2 to store the gradient history for the adaption of the learning rate, i. e. $\mathbf{r}_t = \beta_2\, \mathbf{r}_{t-1} + (1 - \beta_2)\, \mathbf{g}^2$. Finally, Adam [72], standing for "adaptive moment estimation", combines RMSprop with momentum. Hence, the algorithm maintains first moment estimates and second moment estimates of the gradient: $\mathbf{s}_t = \beta_1\, \mathbf{s}_{t-1} + (1 - \beta_1)\, \mathbf{g}$ and $\mathbf{r}_t = \beta_2\, \mathbf{r}_{t-1} + (1 - \beta_2)\, \mathbf{g}^2$. Then, a bias correction is performed as especially the first steps would be heavily biased towards the initial values of $\mathbf{r}$ and $\mathbf{s}$. The bias-corrected moment estimates are $\hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1-\beta_1^t}$ and $\hat{\mathbf{r}}_t = \frac{\mathbf{r}_t}{1-\beta_2^t}$. Finally, the parameters are updated according to $\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{\mathbf{r}}_t}} \odot \hat{\mathbf{s}}_t$. Throughout this work we will train our deep neural networks using the Adam optimizer, unless stated otherwise.

Before we can start optimizing (or training) our parameters they need to be initialized with some values—another crucial aspect of deep neural networks. The *initialization* can decide upon the convergence of the training process and the generalization of the model to unseen data. Typically, deep neural networks are initialized at random to ensure that nodes which are connected to the same input (and have the same activation function) are initialized with different values. Otherwise these nodes will always receive the same updates using a deterministic training algorithm. In addition, the boundaries of the random values are important. While larger values help to avoid a vanishing gradient as larger weight values yield larger activations, overly large values might again lead to exploding gradients. The *Glorot* initialization suggests drawing the initial values from an uniform distribution $U\left(-\sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}\right)$, while the *He* initialization simple suggests using $U\left(-\sqrt{\frac{6}{n_{\text{in}}}}, \sqrt{\frac{6}{n_{\text{in}}}}\right)$ including the ReLU in their analysis [23]. The bias parameter are usually initialized to be zero, except for the output layer where it can be beneficial to initialize the biases to reflect the biases in the target distribution. Despite the initialization being a hard problem, it was found that CNNs tend to learn similar edge and color blob feature detectors in their early layers. This brings the advantage that once we have a working deep neural network we can use its parameters as an initialization for a new deep neural network. This is often referred to as *transfer learning*. The tasks, however, which these models have to solve need to be sufficiently similar. Furthermore, we need to reduce the learning rate on the transferred layers to avoid destroying the contained information.

Finally, with the large amount of free parameters and the concomitant huge capacity of deep neural networks, overfitting becomes a serious issue. Therefore, the training process requires good regularization strategies. One of them, the dropout, was already explained in the previous section. Another approach is to directly address the parameters (outlined in Figure 2.10b). The idea of the *L_2-regularization*, for example, is to continuously push the parameters towards zero as long as they do not receive any updates which clearly push them in another direction. This further keeps the weights in a numerically stable float range so that there are no large differences in the order of magnitude between input, activations and parameters. Finally, *data augmentation* is a common regularization technique. It describes the artificial extension of the training set by transforming the existing training samples [71]. Data augmentation includes all kinds of transformations, ranging from rigid

transformations, like rotating and mirroring the data samples, over affine transformations, like scaling and shearing, and arbitrary elastic transformations up to all sorts of color transformations, like contrast variations, color shifts, and the introduction of artificial white noise. Furthermore, the creation of augmented data samples can be learned by a generative model. This, however, goes beyond the scope of this introduction and we refer to [73] for more details.

Object Detection. Knowing how to build a deep neural network and how to train all the free parameters of the model, we can now deal with some task-specific network architectures to find out how to solve our problems. First, we survey the mere localization of our target structures (i. e. object detection) as the localization can be further combined with a more precise segmentation mask in a second step. Typical representatives are the faster region-based CNN (Faster R-CNN) [74] and the you only look once (YOLO) [75] architectures which are inspired by more traditional voting approaches.

The approach of the Faster R-CNN is to utilize the features computed by the last convolutional layer of an image classification network as input for a region-proposal network (RPN) which is trained to predict K candidates for bounding boxes and their probability of being part of an object. The RPN is implemented as a three-layered network. First, a convolutional layer with a kernel size of $k = 3$ computes some intermediate features from the input feature map obtained by the classification network, then two convolutional layers with a kernel size of $k = 1$ predict the bounding box candidates and the probability of belonging to an object in parallel. After that, the features from the last convolutional layer of the image classification network within a given candidate region are pooled to a fixed size and serve as input for the final prediction. Each candidate predicts a bounding box and a corresponding object class. Finally, a non-maximum suppression helps to clean up the results. This model further can be augmented to predict a segmentation mask for each object. The Mask Faster R-CNN [76] does so by predicting C binary segmentation masks—one for each of C possible classes—for each of the candidates. The predicted segmentation mask, however, has a fixed size: the size of the pooled input region. Because of the necessity to up-sample segmentation mask to match the original size of the region the result sufferes from a certain lack of details, especially for larger objects.

The approach of YOLO is similar, but instead of using a RPN to create candidates the input image is subdivided into $S \times S$ patches, each of which votes for K candidates for object bounding boxes, a confidence value and an according class label. Therefore, we end up with way less candidate bounding boxes of which most of the irrelevant ones have a low predicted confidence so that the non-maximum suppression can be omitted.

Even though these architectures have been tested successfully on images of crowded scenes with plenty of objects [76], they can only detect a limited number of objects at once, which renders them unsuitable for our task. We cannot limit the number of detectable defects because we do not know in advance if there are only a few large defects in a CT scan or many smaller ones. We also cannot arbitrarily increase the number of predicted candidates because each additional candidate vector further occupies valuable space on the GPU. For theses reasons, this type of architecture will not be considered in this work. Instead, we go straight for the semantic segmentation and compute the position and extent of individual instances from the segmentation mask.

Semantic Segmentation. The principal idea behind the various segmentation architectures is to generate a latent feature representation of the input and to then up-sample the results to the original size of the input to obtain a per-pixel segmentation mask. In contrast to the object detection architectures which yield a given number of bounding boxes and class labels per forward-pass, the segmentation architectures yield a class label for every pixel in the input which leaves us with C segmentation masks (one for each of C classes). The determination which pixel belongs to which instance has to be done in a post-processing step. Typical representatives are the FCN [12], U-Net [18], and SegNet [77].

The concept behind the FCN is that any classification architecture, which was pre-trained, for instance, on ImageNet, can be turned into a segmentation architecture by converting the fully connected layers into convolutional layers and placing the pre-trained weights of the fully connected layers in the convolutional layers. This transfer process is often referred to as network surgery. As the pooling layers reduce the spatial resolution, the predicted results have to be up-sampled to match the size of the input. In the FCN this is done with transposed convolutions. In [12] two architectures are compared: (i) one which directly up-samples the output of the converted classification layer to the full image size, the FCN-32s (as a 32× up-scaling is necessary) and (ii) one which includes additional information, "Combining what and where" [12], by up-sampling the classification result step-by-step and adding additional predictions, which are inferred from the output of the pooling layers with corresponding output size, to the intermediate results before up-sampling them again. This model is referred to as FCN-8s (as not all pooling layers are used to add information and the final up-sampling is an 8× up-scaling). The FCN-8s is found to perform significantly better, capturing more details in the segmentation mask. The great benefit of this approach is that only the layers which up-sample the prediction have to be trained. That means only a smaller amount of precisely labeled training data is necessary, as the down-stream part of the network is trained on a classification data set. A similar approach is taken by DeepLab [13]. Here, however, the result of the converted classification layer is up-sampled using a bi-linear interpolation and then refined using a fully connected conditional random field (CRF) [13]. The CRF ties the coarse classification results to the pixel boundaries in the image.

U-Net and SegNet take this procedure to the next level: They utilize symmetric encoder-decoder-pairs, where the decoder contains the same amount of layers as the encoder and each layer in the decoder has the same number of channels as its corresponding layer in the encoder. For each pooling layer in the encoder there is a corresponding un-pooling layer in the decoder. So called *skip-connections* provide the decoder with the necessary detail information by concatenating (or, in case of the SegNet, adding) the features with the same level of detail of the encoder to the up-sampled results of the decoder. Consequently, the network has a lot more parameters which have to be trained from scratch when coming from a classification architecture and, therefore, need more precisely labeled training data.

Nevertheless, the increased up-sampling effort pays off: These models significantly improve the prediction performance for common image scene understanding challenges like PascalVOC [61] or Cityscapes [17] and are at least a part of all state-of-the-art segmentation pipelines. Furthermore, these models have been successfully used on three-dimensional medical data sets [78], for example, to segment tumors in the brain [79], the liver [80], or the lung [81]. Hence, these architectures seem to be a promising starting point for our purposes.

2.3.2. The Pre-Deep Era: Traditional Methods

Traditional methods, in contrast, usually follow the approach described in the beginning: First, we need to select distinctive candidate regions from salient points in the data. From these candidate regions and their surroundings, we extract specific hand-designed features. Finally, we use a suitable, "shallow" machine learning model that tells apart the different classes and filters false positive responses from the candidate selection process. The great benefit of traditional approaches is that they are designed to be explainable. Because the extracted features usually are engineered to resemble ideas of human vision and to capture typical properties of the target structures, these approaches should not be neglected. To see if deep learning is really the better choice, we need to form a sound baseline. The actual algorithms forming the baseline are explained in Chapter 4. Here, we focus on the necessary basic concepts.

Candidate Selection. The purpose of the candidate selection is to propose possible instances of the target object in the data for further consideration. Therefore, the candidate selection algorithm needs to scan the entire input. This requires that the algorithm is able to process a large amount of data in a reasonable time. Furthermore, the candidate selection algorithm needs to be quite sensitive, finding every possible instance of the target object. Each instance which is not found in this stage remains undetected. Simple filter-based image processing techniques, for example, are promising candidate selection algorithms. They are able to quickly scan the input data for *salient points* of a given size. Salient points such as edges and corners in particular are usually characterized by a high content of information, which arises from a large local variance in gray-values and renders them less prone to noise. Homogeneous regions, in contrast, show only little to no variance in their gray-values. Hence, the effect of noise in these regions is much stronger.

The property that there is a large variance in gray-values near salient points is exploited, for instance, by the Harris corner detector [82] and the blob detectors which are described by a difference of Gaussians (DoG) [83]. The latter comes in handy especially in the detection of defects, which more or less have a spherical outline that match the DoG filter. To directly detect more complex objects we can use template matching approaches which correlate entire objects (or their individual parts) with the input data [84–86]. Here, it is often helpful to operate on more abstract representations of the image to find suitable candidates. For instance, we can use an edge image in order to eliminate disturbing influences of probably unnecessary information like specific colors or the illumination of the image.

In the case of semantic segmentation we further need to propose a segmentation mask in each candidate region. Depending on the candidate selection algorithm we can derive a segmentation, for example, by applying a threshold to the filter responses or by growing the candidate region starting from a seed point which we determined from a non-maximum suppression of the filter responses. Here, we refer to [82] and [87] for more details on the algorithms.

Feature Extraction. The features serve the purpose of (a) describing the target object and (b) to tell apart different objects from one another or a cluttered background. Furthermore, it usually cannot be assured that the target object always has the same orientation or size within the data. A good feature, therefore, not only has to be distinctive, it also needs to be invariant to changes

in scale, rotation, translation, and even illumination. More distinctive features make a classification easier and allow for simpler classifiers. Such high-level features are usually more object-specific and tailored towards particular tasks. In contrast, low-level features like gray-value, color, or gradient statistics [88–91] are more general and applicable to a broad variety of tasks. This requires the classifier to be trained towards properly combining the low-level features for a reliable decision. For more details on how to compute different feature we refer to [82]. In this work we deal with rather simple structures. Hence, we stick to rather simple features which comprise gray-value and gradient statistics.

Classification. The final step is to make a decision about each candidate whether it is part of the target class or not. Therefore, we have to choose a classification model from the huge machine learning toolbox, which serves our needs best. Here, we refer to [92] for more details. One example is the random forest [93], which is explained in more detail down below. In the field of defect detection we will explore the approach the traditional object detection pipeline described in [49] in Section 4.2.1, which uses a random forest to classify defect candidates based on their gray-values and their curvature.

Pixel-level Classification. A slightly different approach which is more reminiscent of the end-to-end strategies of deep learning approaches is the pixel-classification described in [94]. Instead of extracting features from pre-selected instances, we extract the features for each and every pixel in the image. To contain the run time these features are usually less complex than those extracted for a few individual instances. Furthermore, in this case we do not yet have a subdivision of the image which would allow us to tailor the extracted features more towards the object we actually like to find. Examples for features which can easily be extracted for each pixel are filter-based features like gradient filters. More on them in Section 4.2.2. These features are not only applicable to defect detection but to the analysis of all kinds of images. However, it is important that the extracted features cover fine structures, for a precise segmentation, and the entire target object, for a proper recognition.

Here, we focus on the classification step: A *random forest* assigns a class label to each pixel based on the densely extracted features. The random forest is an ensemble method using bootstrap aggregation (*bagging*). The principal of bagging is to use the mean prediction over a set of strong bootstrapped classifiers. This reduces the variance of the results of the individual (noisy) classifiers [92]. This approach contrasts to boosting in terms of (i) the classifiers each contributing equally to the results, (ii) the individual classifiers being strong classifiers (it is not sufficient that they are barely better than guessing), and (iii) the classifiers being independent of one another. That means each *decision tree* T_b of the random forest, which in total consists of B trees, is trained with randomly chosen samples of the training set. The probability $\hat{y}$ of a pixel belonging to a certain class is the mean of the results of all T_b which make their decision based on the features $\mathbf{x}$ (see Equation (2.5)). Alternatively, for a hard class assignment we can use a majority vote.

$$\hat{\mathbf{y}} = \frac{1}{B} \sum_{b=1}^{B} T_b(\mathbf{x}) \tag{2.5}$$

The training algorithm of a random forest is straight forward: For each tree in the ensemble we first bootstrap-sample, i. e. sampling with replacement, a set of n feature-label pairs $(\mathbf{x}, \mathbf{y})$ from the complete training set (where $\mathbf{x}$ is a vector of m features and $\mathbf{y}$ contains the according class assignment). Then we grow a decision tree based on the sampled training data by (i) randomly picking m^* of our m features, (ii) given the m^* features determining the threshold which separates the different classes in the node best, i. e. the best split, and (iii) introducing two child nodes according to the split. We do this until each leaf node only contains elements of a single class, i. e. a pure node, or an early stop criterion is fulfilled. These criteria act as regularizing hyper-parameters and are explained below. First, we need to define how to determine the best possible split. Our goal is to minimize the impurity in the child nodes. That means we are looking for the best split point s along our m^* features that minimizes our cost criterion Q in both child nodes R_1 and R_2: $\min_s [Q(R_1) + Q(R_2)]$. As cost function we can use, for example, the cross-entropy (see Equation (2.6)) or the Gini index (see Equation (2.7)), where p_i is determined by the portion of instances with class i in the region. Both are differentiable and, therefore, suited for optimization methods. Furthermore, compared to using a misclassification error as cost function, both cost functions prefer the creation of pure nodes.

$$Q_{\text{x-entropy}} = -\sum_{i}^{C} p_i \log p_i \tag{2.6}$$

$$Q_{\text{Gini}} = \sum_{i}^{C} p_i (1 - p_i) \tag{2.7}$$

To tune the random forest, we can, for example, vary the number of trees in the ensemble and define a maximum depth of the decision trees, i. e. a maximum number of splits we have to compute to receive a decision. Both parameters have an influence on the complexity of the model and directly affect the prediction time. However, there are more hyper-parameters which can be tuned when training a random forest. These have a regularizing effect as well but only indirectly affect the prediction time: (i) We can choose the maximum number of leaf nodes which limit the size of the trees but allow for very deep unbalanced trees, (ii) we can define the minimum number of samples in a leaf node allowing impure leaf nodes in the final decision tree, and (iii) we can define a minimum necessary decrease in impurity for a split to be made. These hyper-parameters can be tuned, for example, by using a random search. In our case, we found it most beneficial to focus on the number of trees and their maximum depth.

With that, we have introduced the necessary theoretical background and discussed the relevant related work. Further details will be introduced when necessary. In the next chapter, we ask the question how we can acquire the necessary amount of training data to properly train our machine learning algorithms for the detection of defects in CT scans of cast aluminum parts.

3. The Quest for Data

The supervised training of machine learning models requires labeled training data from which they derive (or "learn") the values of their free parameters. This training data contains samples drawn from distribution of data in the real world, from which the models build up a statistical abstraction from reality, which is then used to extrapolate to unseen data. The more data, i. e. the more samples of the real data distribution, they have at their disposal, the better this abstraction becomes, as it is more likely to cover all the different aspects we encounter in the real world. The actual amount of data which is necessary to build a good abstraction also depends on the number of free parameters a particular model has: The more free parameters are available, the more training data is necessary. Furthermore, using more data prevents the model from simply learning its training data by heart and increases the chances that the model grasps more relevant aspects of its task. Especially deep learning models have a large amount of free parameters, in the order of magnitude of hundreds of thousands or even millions. Therefore, they crave vast amounts of labeled data for a proper *supervised training*. This leads to an increased interest in semi-supervised and unsupervised methods to reduce the necessity for large amounts of labeled data [95–98].

For many tasks there already exist ever-expanding labeled data sets: For image classification, for example, there are ImageNet [59, 99], PASCAL VOC [61], and Microsoft COCO [60] with hundreds of thousands of images of thousands of categories. They contain objects of all shapes and sizes ranging from small insects over mammals and plants to all kinds of man-made structures like cars, airplanes, and ships. Even for more specific tasks there are data sets. For example, for road scene understanding we have CamVid [100, 101], KITTI [62], Cityscapes [17], and many more to train our models. These data sets contain labels of different complexity ranging from a simple distinction between road and background to a detailed instance segmentation of pedestrians, cars, traffic signs, and so on. In the field of three-dimensional volumetric data there are, for example, several medical data sets like BraTS [79], SLIVER [102], LIDC-IDRI [81], or LiTS [80]. They contain CT scans (or magnetic resonance imaging scans) with labels that either outline individual organs or tumorous tissue in the image.

When looking for an appropriate data set for the detection (and segmentation) of defects in CT scans of cast aluminum parts, however, things become much more difficult. In the field of non-destructive testing there is a great silence regarding data. This can have various reasons: (i) Compared to road scene understanding and medical image processing it is not as publicly present. (ii) The amount of data produced by industrial CT quickly exceeds the terabyte-scale. Gathering the amount of data which is necessary to properly train a deep learning model is quite expensive—especially because labeling requires the knowledge of domain experts. (iii) Finally, CT scans reveal everything about the part under examination. This raises concerns about the intellectual property of the manufacturer. Nevertheless, for radiographic tasks, there is the GDXray [103] data set. It contains data for a broad variety of applications: security checks of luggage, quality checks of food, the inspection of welds,

and even the inspection of cast aluminum parts. But, for our task we require CT scans which cannot be computed from the single radiographs in this data set.

If the available labeled data is not sufficient to properly train a model, often *synthetic data* is called in. For example, there exist various approaches to generate synthetic data for road scene understanding like Virtual KITTI [104] which resembles the data of KITTI in a simulated world. This allows to easily vary the conditions of weather and daylight in the images. Another approach uses existing realistic simulations to generate synthetic training data: video games [105]. Simulations are also used in models which made it to production-level. For example, the pose estimation model which is used in Microsoft Kinect is trained with the help of synthetic depth maps [106]. A different example is the gaze estimation model developed by Apple. They use images of synthetic eye models which come with a precise ground truth and refine them with adversarial training to look even more realistic [107]. Another field of application which heavily exploits simulations and synthetic data is the training of autonomous robots and drones via reinforcement learning. Here, training in a simulated environment has the huge benefit of avoiding actual damage on the hardware, i. e. the robot or the drone. However, transferring the trained models to the real world is challenging [108–111]. The big problem with synthetic data is the (sometimes large) gap between the domain of real images and the domain of synthetic images. This domain gap is often referred to as the "reality gap" [109]. As a result, synthetic data is often used for pre-training only. Nevertheless, the more realistic the synthetic data is, the better the results.

Without labeled data at hand, we have to come up with our own data set. In this chapter we, therefore, explore two different ways to obtain annotations for real data: the sparse annotation of individual voxels by looking at their neighborhood and the creation of dense annotations with the help of high-quality CT scans. Furthermore, we explore three different approaches to generate realistic synthetic data, which promise to bridge the "reality gap". These are domain randomization, generative models, and realistic simulations. Finally, we develop an automated simulation pipeline which allows us to generate an arbitrary amount of precisely labeled training data.

3.1. The Challenge of Labeling Real Data

Labeling the data for defect detection in CT scans of cast aluminum parts is considerably harder than labeling images for image classification or road scene understanding. Most notably, this is because (i) we have to deal with three-dimensional *volumetric data*. This type of data is particularly hard to grasp, as we have to go through individual slices in all three dimension to get the necessary contextual information. (ii) Additionally, we are looking for structures *without* an *inherent shape*. Due to their daily lives, humans are more familiar with road scenes than they are with casting defects. No pedestrian quite looks like another, but that doesn't hinder humans to recognize them. Defects, in contrast, have an arbitrary shape and are rarely seen. Nevertheless, the shapes of different types of defects follow different rules, but it requires the knowledge of domain experts to recognize them. (iii) Moreover, the sometimes heavy *image artifacts* which arise from scatter, beam hardening, and insufficient exposure times further impede the labeling process. This demands an increased concentration from the experts and contributes to their exhaustion (which also is an argument for an automated inspection). When labeling medical images we encounter similar problems, e. g. the labeling of tumorous tissue. Here, the spatial resolution and the contrast sensitivity are further limited

due to the reduced dose a patient can bear. We now explore two labeling strategies to ease the effort for our experts.

3.1.1. Sparsely Annotating Real Data

The first approach is to sparsely label single voxels whether they are either part of a defect or of a structural loosening or are not part of a flaw. To ease the labeling process for our experts, we develop a small application with a simple web interface (see Figure 3.1a). Here, the experts label the CT scans voxel by voxel with a simple click on one of the "Decision"-buttons. To provide our experts with contextual information we show a neighborhood of 33×33 voxel around the current target voxel by means of the axis aligned slices. In addition, we mark the target voxel with a red cross. Our experts then have to decide which class the voxel belongs to: "defect", "structural loosening", or "no defect". If they are not sure, they are allowed to mark the voxel as "unclear".

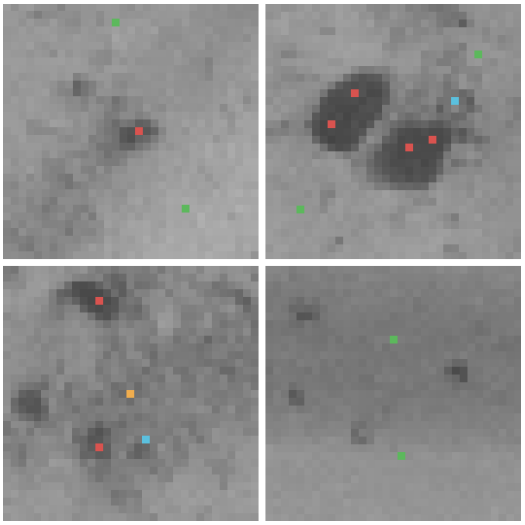
The volume data which we obtain from each CT scan ranges in the order of magnitude of a billion voxels. Labeling every voxel in each volume would not be feasible. Furthermore, the data only contains few defects so that selecting voxels which belong to a defect by randomly choosing voxels from the volume is rather unlikely. Hence, we use an existing filter-based defect detection algorithm to provide us with an initial guess where to find defective voxels. Then, we especially sample close to these candidates and add only a few random voxels from the rest of the volume. To obtain information about as many defects as possible, we configure the candidate selection algorithm to have a high recall. Due to the high recall we also include many artifacts and other characteristics of a CT scan which easily are confused with defects. Some examples of the sparsely labeled CT scans are shown in Figure 3.1b. We assign a label to each voxel, as long as at least 66 % of our experts agree in their decision. Otherwise we assign the label "unclear".

We abandon this approach because of several severe drawbacks: (i) This procedure requires too much time to generate a sufficient amount of labeled data. (ii) Particularly interesting for the training of machine learning algorithms are the corner cases, for example, at the transition of material and defect. This information is necessary to obtain a crisp segmentation later on. However, the voxels in these regions are rarely labeled and if so, the labels are ambiguous. (iii) Many voxels are labeled as "unclear" because the necessary context is missing. The experts cannot be sure if the voxel belongs to a defect or a structural loosening. In these cases it could be beneficial to "scroll" through the data, taking the preceding and subsequent slices into account. This is a deficiency of the labeling tool. (iv) Even though our candidate selection process has a high recall, the smallest of the defects seem to be missed out. We see better chances to obtain the necessary amount of training data by improving the creation of dense labels.

3.1.2. An Improved Labeling Process for Dense Labels

Our second approach is to tackle at least one issue which impedes the creation of dense labels: the image artifacts. Having to deal with industrial CT and cast aluminum parts has the benefit that the object under examination does not mind higher doses and does not change unexpectedly over time. This permits the usage of higher energies and longer exposure times when making the CT scan. We, therefore, acquire two CT scans of the same object. A *high-quality scan* with as little image artifacts

(a) The web interface of the labeling tool to create sparse annotations for real CT scans. It shows the three axis-aligned slices centering at the voxel which is to be labeled, and the possible decisions.



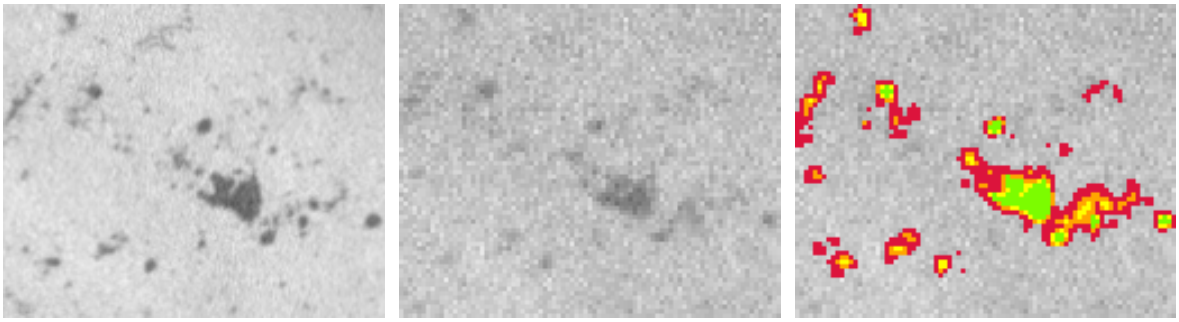
(b) A few examples of labeled defective and defect-free voxels. We see that we only have very few information per volume, which is not enough to train a confident and robust classifier.

Figure 3.1.: To ease the process of labeling individual voxels, i. e. sparsely labeling the CT scans we have, we develop a small web interface that allows our experts to click through the data set.

as possible (see Figure 3.2a) and a *"normal" scan* which holds all the challenges for training (see Figure 3.2b). Now, the idea is to densely label the high-quality scan and transfer the annotations to the "normal" scan which is used for training later on.

We hope that the better the high-quality scan is, the better the labels become. If we acquire a CT scan in which thresholding the gray-values already yields a pretty good segmentation, it should considerably ease the task of labeling. Compared to the "normal" scan, the high-quality scan has a higher spatial resolution and a much higher contrast-to-noise ratio (CNR), i. e. lower noise-level. This requires very long exposure times. The exact scan parameters, however, depend on the part under examination and are provided for the individual parts later on. Having the CT scan of high image quality, we hand them over to our eight experts and ask them to label each voxel of which they think it belongs to a defect—not considering if it could be harmful or not. For this task, we allow them to use all the functionality available in VGSTUDIO MAX (Volume Graphics GmbH) as it is the software they are used to work with.

Now, we need to combine the individual labels of our eight experts and transfer the result to our CT scans of "normal" image quality which are more artifact-afflicted and have lower spatial resolution. The challenge here is the diversity of the labels produced by our experts: Some label single voxel defects in the high-quality scan, others were labeling not as accurately and not as precisely. Some try to include information about how much of a voxel belongs to a defect yielding a soft transition between material and defect, others create a binary mask which only coarsely includes the defect. Each mask is stored as 8 bit unsigned integer volume data with a maximum value of 255. That means the binary labels either assign 0 or 255 to a voxel and the smooth transition of the other labels lies somewhere in between. First of all, we remove instances which contain less than 8 voxels as they are not visible in the "normal" scan. Then, we add up all labels, resulting in a mask with a maximum



(a) A patch of the hiqh-quality scan with high spatial resolution and low noise-level for labeling.

(b) A patch of the "normal" scan for training. It contains all the challenges we encounter in the quality laboratory.

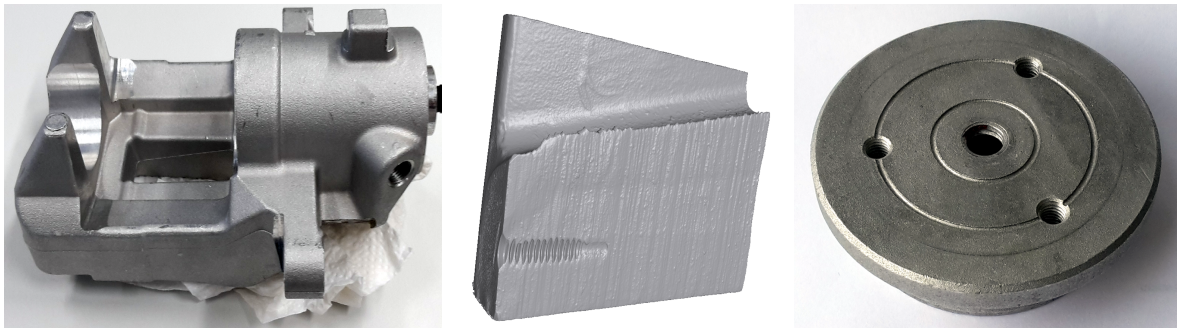
(c) Different experts label the data differently, leaving us with ambiguous labels ranging from full accordance (green) to only a single expert labeling a voxel (red).

Figure 3.2.: To ease the process of densely labeling entire CT scans, we make two CT scans of the same part: a high-quality scan for labeling (a) and a "normal" scan for training (b). However, our eight experts have different opinions about what a defect is, which leads to ambiguous labels (c). Especially the actual extent of a defect remains unclear and varies from defect to defect.

value of $8 \cdot 255 = 2040$. Using this value as confidence, we can compute the mean confidence and the maximum confidence for each labeled defect. If the mean confidence is much smaller than the maximum confidence, we have an instance where our experts are particularly indecisive. Here, it is helpful to do a binary erosion of the label mask, as the difference might result from a too coarse labeling of an individual expert. Finally, we remove all instances which were not labeled by at least two experts, or are smaller than 8 voxels after performing the erosion. We store the combined ground truth again as 8 bit unsigned integer volumes, linearly mapping the values to the range $[0; 255]$.

Having the combined ground truth of all of our eight experts, we need to transfer it from the high-quality scan to the "normal" scan. Unfortunately, the corresponding scans were not necessarily done right after another, so they differ in position and orientation. However, for the transfer we need them to be aligned precisely. Therefore, we use the "advanced surface determination" of VGSTUDIO MAX to compute the precise material boundaries of the object in both CT scans. Then, we use the "registration algorithms" of VGSTUDIO MAX to align both CT scans guided by their material boundary. Finally, we re-sample the ground truth mask of the high-quality scan to match the voxel grid of the "normal" scan.

Despite all the effort of making high-quality scans, the labels remain *ambiguous* (see Figure 3.2c). Furthermore, some of our experts label every tiny defect, while others only consider instances that have a minimal size. Some of our experts tend to label a structural loosening as defect, some label only "darker parts" of a structural loosening as defect, and others do not consider a structural loosening as defect at all. This could be explained by using different monitors which display the same gray-value at a different intensity, but even worse is that there are examples of structural loosening in the data set that none of our experts consider as defect—not even those who label other occurrences of structural loosening. The combined label mask we obtain allows us to take the accordance of our experts into account. The more our experts agree on a voxel being part of a defect the higher the value in the combined label mask is. The problem that follows for the process of training is where to draw the line? What should we tell our machine learning method about what actually is a defect? Even when labeling objects of our daily lives the boundaries are not really clear [28]. In med-



(a) A break caliper of a rear wheel of a car.

(b) A material sample taken from a larger aluminum part.

(c) A pivot cap of a laser scanner.

Figure 3.3.: The cast aluminum parts which we scan for our data set of real CT scans. We make two CT scans of each part: a high-quality scan for labeling and a "normal" scan which holds the challenges for our algorithms.

ical data we encounter the same problem. For example, when annotating tumors in the lung [29]. Here, the usual procedure is to take every voxel as part of the target class that is labeled by at least one expert and rather model a probability distribution of being labeled [29]. The idea is to provide the specialist who has to cross-validate the results with an indication of every possible incidence and to give him a notion of what other experts would think about this incidence and its extent. For evaluation, the result is not compared against a binary ground truth but is evaluated on how well the predicted distribution matches the probability distribution given by the experts, e. g. by calculating the generalized energy distance [112]. This approach accepts higher false positive rates and the higher run time which is necessary to generate the samples for the estimated probability distribution. We, however, need a precise segmentation of each defect instance which is in the part under examination instead of a probability distribution, in order to compute derived measures like its volume, compactness, or sphericity. We need this information to draw further conclusions about its associated fragility and criticality.

In a first run, we make a high-quality and a "normal" scan of three cast aluminum parts: a brake caliper of the rear wheel of a car (see Figure 3.3a), a material sample of a larger cast aluminum part of a car (see Figure 3.3b), and a pivot cap of a laser scanner (see Figure 3.3c). The break caliper has a size of 150 mm × 140 mm × 80 mm and is scanned with 295 kV at 320 µA and a filter of 1.0 mm copper. The material sample has a size of 80 mm × 120 mm × 70 mm and is scanned with 295 kV at 320 µA and a filter of 1.0 mm copper. The pivot cap has a size of 80 mm × 80 mm × 40 mm and is scanned with 180 kV at 160 µA and a filter of 0.5 mm copper. The high-quality scans were reconstructed with a voxel size of 41 µm from 2000 projections with a total scan time of 120 min, while the "normal" scans were reconstructed from only 1500 projections with a total scan time of 15 min and have a larger voxel size of 82 µm. Unfortunately, the CT scan of the break caliper was not very useful for our purposes. Firstly, it contains only very few tiny defects and one large structural loosening and secondly, there still was a steel screw in the part during the scan process, which renders severe metal artifacts into the final scan. Thus, for now we only create a ground truth for two of the real cast aluminum parts: the *pivot cap* and the *aluminum part*.

3.1.3. The Crux With Training on Real Data

Combining all annotated voxels from Section 3.1.1, we roughly have 1000 samples of which about 100 are "defective". With this small amount of data we are not able to train a robust model which can precisely segment defects in the broad variety of artifact-afflicted CT scans we encounter in production. The lack of information in the boundary regions of the defects exacerbates this effect. Here, most of the experts either chose "unclear" or contradicted each other. Nevertheless, we give it a shot and train a small CNN for classification. As we only have very few labeled samples, we build the net with only three convolutional layers for feature extraction and a fully connected layer which yields the final decision. For training we crop patches of $32 \times 32 \times 32$ voxel around each labeled voxel and let the net predict if the given patch is "defective" or not, i. e. whether the center voxel was labeled as "defective" by our experts. Later, for inference, we can use a sliding window to predict the likelihood of a voxel being "defective" patch by patch. To speed up the inference we can convert the model to a FCN by converting the fully connected layer into a convolutional layer with a kernel size of $1 \times 1 \times 1$. To obtain a probability for each voxel, we need to upscale the probability map we get from the FCN. Even though we end up with a much smaller probability map which we have to upscale, the results of the sliding window and the naive FCN are pretty similar. However, the results end up being fuzzy and indistinct (see Figure 3.4) due to missing information about boundary regions and the lack of spatial resolution in the decision making layers.

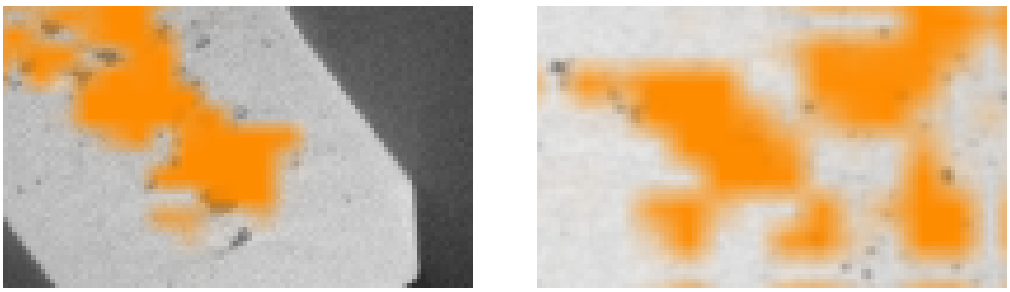


Figure 3.4.: The orange overlay shows the prediction of the trained model. Using the scarce labels from our sparsely annotated CT scans for training, the results become fuzzy and indistinct. This is because we lack precise labels for the more interesting parts, i. e. the boundary regions between defect and material, and have to upscale the results to the full image size.

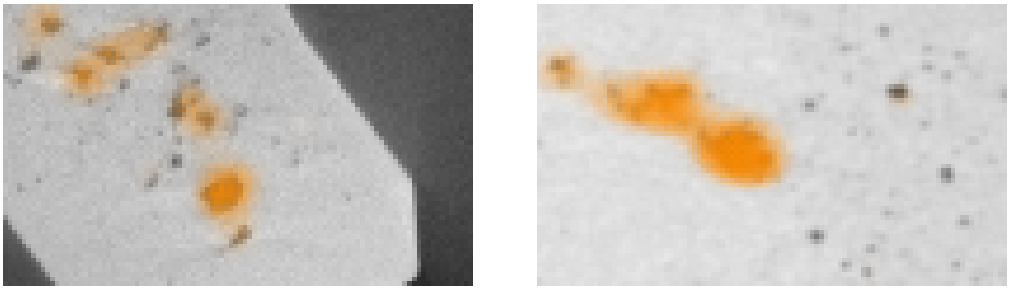


Figure 3.5.: With the densely annotated CT scans we can train a FCN to output a more refined segmentation mask. However, we still miss precise information about the boundary regions of the defects which is reflected by the results. Even though the defects are better located, the segmentation remains fuzzy and misses small defects. Again, the orange overlay shows the prediction of the trained model.

With dense labels we have much more information available and consequently are able to train a more sophisticated FCN which takes the information contained in earlier layers into account while up-scaling the predicted probability map. Even though we use dense labels for training, we do not achieve considerably better results. The defects are located more properly, but we are far from a crisp segmentation of individual instances (see Figure 3.5). Most notably, this is because the labels which we use for training are ambiguous. If we have regions which are labeled as being part of a defect and similar regions which are labeled as "flawless", the error-gradient can be imagined as pointing in one direction or the exact opposite direction, respectively. These disturbances become more destructive when the model does not see the whole training data at once as it is the case when training deep neural networks. Therefore, the trained model cannot yield precise predictions for ambiguously labeled data.

We did not yet consider to use more sophisticated labeling tools which, for example, utilize random forests to enable a guided segmentation by simply annotating the most relevant regions [94]. Such tools often require an extensive introduction and data preparation and expert time is expensive. Instead, we decide to use the two CT scans for which we already have dense labels for validation and move to using synthetic data for training.

3.1.4. Pre-Training With Real Data

But there is another way we can use any real CT scan—labeled or not: The *unsupervised* pre-training of a neural networks as warm-up for their kernel weights and to give them a notion about the data they are going to work with [97, 98, 113, 114]. The idea arises from *transfer learning*. Neural networks which are trained to solve similar tasks tend to learn similar features, i. e. filter kernel weights, in their early layers. Using these weights to fine-tune the model for a new but similar task, we need less training data and the fine-tuned model better generalizes to unseen data [115]. Pre-training utilizes this effect to bring randomly initialized networks closer to their purpose by letting them work with all the available (partially unlabeled) data. The pre-training tasks can, for example, involve the reconstruction of the input data after passing it through the bottleneck of the model, i. e. training an auto-encoder [116]. A variant of this method further adds noise to the input so that the model is trained as denoising auto-encoder [117]. This, however, might be less beneficial for CT data, as it has a high inherent noise-level, which means the denoising auto-encoder never learns to remove all the noise. Another task is the reconstruction of artificially removed patches with help of the remaining data [118–120]. Additionally, we can add more supervision by including meta data into the training process. For CT data, we can use the information about the acquisition of the scan to pre-train our model by letting it predict the distance to the rotation center or the optical axis [114]. Unfortunately, pre-training does not help in our case: ambiguous labels lead to fuzzy results, with and without pre-training. Even though the model which was pre-trained to auto-encode the CT scans is able to reconstruct edges and corners, it yet fails to yield a crisp segmentation mask.

3.2. Of Synthetic Data and Precise Labels

Learning methods, especially deep methods, require a lot of labeled data, which is cumbersome and expensive to obtain. Other fields have proven that utilizing synthetic data can boost the training

process when applied properly [105–108]. For the synthetic data the ground truth can be computed precisely. However, as mentioned before, synthetic data usually does not match all aspects of the real world [109]. Thus, there exist several approaches to bring synthetic data closer to reality, bridging the *"reality gap"*. In this section we will explore (i) domain randomization which arbitrarily alters non-essential aspects of the data [108–111], (ii) generative models which learn how to generate realistic data by trying to fool an expert model [107, 121, 122] and (iii) realistic simulations which use sophisticated ray-tracing methods to come as close to reality as possible [104, 105, 123].

3.2.1. Domain Randomization

When trying to synthesize a large amount of precisely labeled training data in a short amount of time, the resulting data probably is far from reality as too many abstractions in favor of computation time had to be added. We then arbitrarily alter the non-essential configurations of the scene regardless of restrictions from reality, hoping that the trained model is able to generalize to real data [108, 124]. Basically, we create a *superset of reality*. For example, we change the color or texture of the target object or its background, or change some environmental conditions to include all possible—and impossible—combinations [108]. This further reduces the probability that the network learns to recognize an object by its background or a small detail in its texture [125] and guides the model to learn features which we would consider as crucial. Domain randomization is widely used when training real-life agents via reinforcement learning, which requires a lot of trial and error [109,110]. This involves many failures like hitting the wall with the robot or the drone. Therefore, the initial learning has to be done in virtual worlds; doing this in the real world would lead to an expensive hardware abrasion. The challenge is to transfer the actions which the agents learn in the virtual world to the real world, which is basically the same we would like to do. This is where domain randomization comes into play: If these agents are able to navigate through a superset of reality, they should be capable of navigating through reality without expensive adaptations [109, 110, 124, 126].

In case of performing domain randomization for CT data, we start off by creating simple gray-value grids which contain our essential information. We would like to detect defects in cast aluminum parts. Thus, we need a material shape which is represented by a high gray-value in the CT scan, surrounded by air which has a low gray-value. Inside the material we place our defects which again have a low gray-value. It is crucial that the defects have a lower gray-value than the material in which they lie, but they do not need to have the same gray-value as the surrounding air. Then, we apply our domain randomization modifications consisting of (i) a Gaussian filter to blur the data (which corresponds to a variation in the focal spot size), (ii) superimposing linear and spherical gray-value gradients to the data at arbitrary positions (which corresponds to the cupping artifacts introduced by the effect of beam hardening), (iii) adding white noise to image (which emulates to the Poisson noise we encounter in the individual projections of real CT scans), and (iv) rendering dark streaks and rings into the data in arbitrary directions (which corresponds to ring artifacts and partial volume effects). To generate the initial gray-value grids, we sample the meshes from Section 3.3.1 and then randomize them by applying our parametrizable modifications (see Figure 3.6). This allows us to render more interesting defect shapes in the data than mere spherical blobs.

The great benefit of synthetic data is that it comes with *precise ground truth*. If we now use this data and domain randomization for training, the results look promising and we obtain the crisp segmentation map we like to have (see Figure 3.7). Nevertheless, the trained model fails to detect

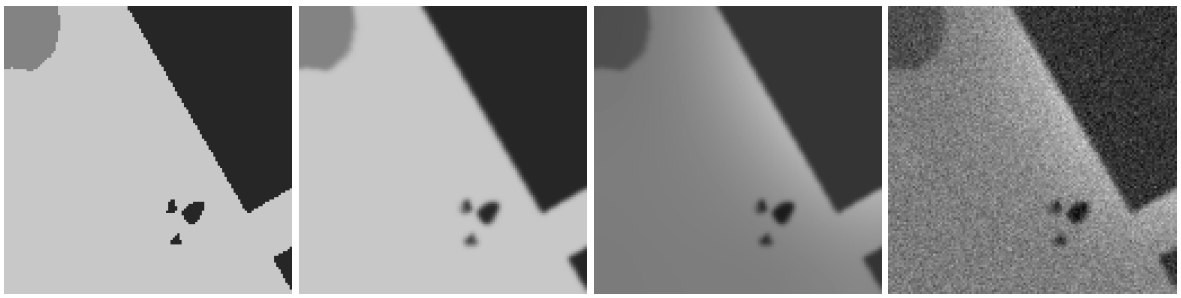


Figure 3.6.: We begin with a simple rendering of a defective aluminum cast part as gray-values (left image) and randomly add modifications to that rendering (right images). We blur the data, superimpose gray-value gradients, add noise, and so on. Note that all four images are used for training, even the unmodified rendering.

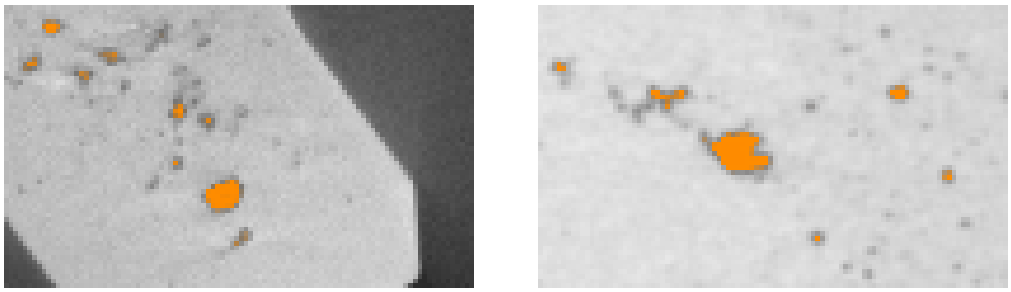


Figure 3.7.: When training with our simplistic synthetic data set and domain randomization we are already able to achieve a crisp segmentation mask. Yet, the model misses small defects and does not generalize well to the broad variety of different CT scans of cast aluminum parts we encounter in production. The orange overlay shows the prediction of the trained model.

small defects and does not generalize well to real data. For example, it is not able to cope with higher artifact levels. The approach of domain randomization has another drawback: The generated training data is not necessarily explainable to domain experts, which, however, is necessary to build up the trust in deep learning technologies among the NDT community.

3.2.2. Generative Models

Our next approach is to train a neural network to close the "reality gap". To tell this *generator* model G how it has to manipulate our synthetic data to look more realistic, we first train a *discriminator* model D which learns to distinguish between real and synthetic data. Then we train G to fool D, i. e. to produce images where D cannot tell whether they are real or synthetic [127]. As it is possible that G simply discovers a small detail which is sufficient to lead D up the garden path instead of producing more realistic images, we iterate the training process: We update D to again distinguish between images which are generated by G and images which are taken from real-life examples. Additionally, we show D generated images from previous iterations to prevent G from altering between two states instead of advancing towards more realistic images. To keep the precise labels valid, we further add the L1-norm of the difference between synthetic input and refined image as constraining loss. This type of generative adversarial network (GAN) is, for example, used to improve synthetic data for eye tracking or gaze estimation [107]. Figure 3.8 shows an outline of the basic training setup. By

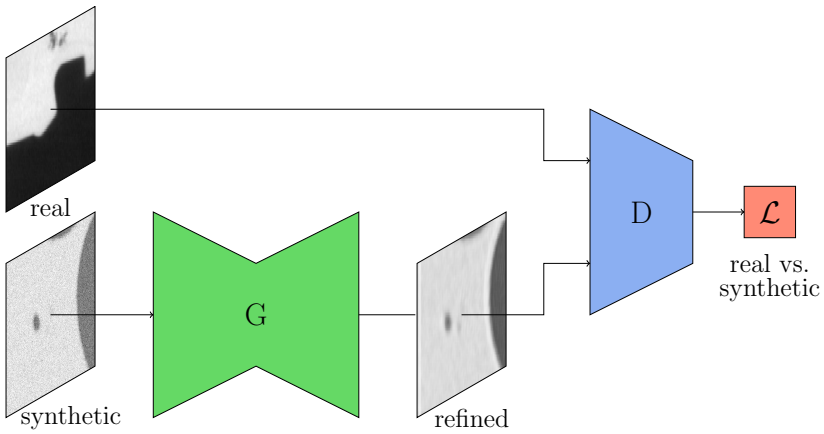


Figure 3.8.: The very basic training setup of a generative adversarial network which allows us to refine our synthetic data, for which we have a precise ground truth, to look more realistically. The generator G modifies the synthetic input image and tries to fool the discriminator D which has to distinguish real samples from synthetic ones.

adding Gaussian noise to the synthetic input image, we enable G to render more realistic structures into the wide flawless regions and the background, which otherwise would have all the same gray-value and, therefore, the same appearance in the refined image. As loss function $\mathcal{L}$ we use a simple Wasserstein metric [128] which is also known as "earth movers distance", i. e. how much does it cost to transform one pile of dirt (or in our case a probability distribution) into another.

In the end, we can use the refined images generated by G with the precise labels of the synthetic data to train our segmentation net. Figure 3.9 shows some examples of the refined images. Unfortunately, the images only look realistic on first sight. They do not stand up to closer inspection. When comparing the input image to the refined images, we notice that the defects in the refined images become much larger. Furthermore, the generator seems to have a hard time generating images with a fine-grained noise distribution. This might be because the variation in the gray-values are more subtle for noise than for edges in the image. In addition, more complex types of artifacts,like ring or streaking artifacts are missing, too, due to the limited receptive field of the generator. While these images might fool the layman, to the domain expert it is pretty obvious that they are fake. GANs are part of a wide and open field of research and the proper training of a GAN is quite hard. This leads to a vast amount of different architectures with different auxiliary loss functions which have been used to avoid exploding gradients, drifting into the generation of unrealistic images, and other issues. One example is the cycle consistency loss introduced by the CycleGAN architecture [129, 130]. Here, another generator tries to reconstruct the original image from the synthetic one. Another challenging aspect when training a GAN is the *mode collapse* [131–133], i. e. the generator producing a limited sample variance only. It might focus only on a small detail which is currently necessary to fool the discriminator: If we train a GAN to produce images of handwritten digits, it might be that the generator only produces sevens because for the current state of the discriminator they could be particularly hard to distinguish from real examples. With the adaptation of the discriminator to detect the generated images, the generator might then start to meander between different modes, never serving all the possible modes [132]. While using the Wasserstein metric as loss function limits the possibility of mode collapse, it does not entirely prevent it. Instead, the weight clipping which is used to enforce the Lipschitz constraint also limits its capability [134]. Therefore, a growing num-

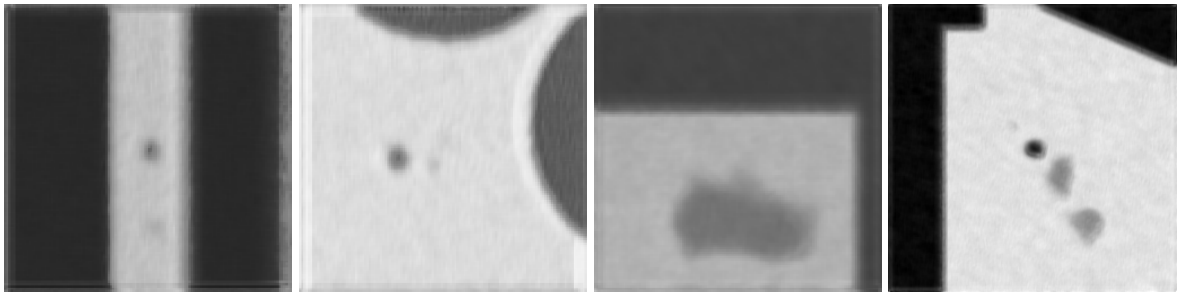


Figure 3.9.: The refined images produced by the generator look more realistic at first sight. However, a closer look reveals two major problems: The defects become noticeably larger compared to the original mask which is used as input for the generator and all the generated images lack the fine-grained noise of typical CT scans.

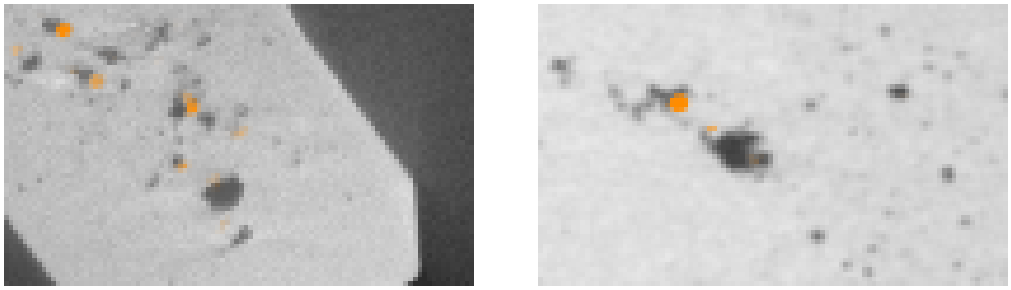


Figure 3.10.: The results of the segmentation net which is trained using the refined samples of the generator are not as good as we expect them to be. Most notably, the model is way more restrained in labeling defects and the labels are slightly off. Furthermore, it struggles with larger defects. The reason could be that the refined images do not match with the labels anymore, even though we prevent changes by penalizing inconsistent transformations. The orange overlay shows the prediction of the trained model.

ber of approaches try to prevent the mode collapse, for example, by adding different regularization losses and constraints to the network, while keeping the full model capability [134–137].

As the generator changes the appearance of the defects, the labels do not correspond as precisely to the defects anymore. Even though we use a consistency loss, the defects often appear to be larger and shifted by a few voxels so that the labels are slightly off. We also observe this in the results of the segmentation model which is trained with the refined images (see Figure 3.10). We only find parts of the larger defects and the predicted labels are slightly shifted to one side of the defect. Moreover, we find that the segmentation model has much more trouble in dealing with noisy CT scans. To improve the results it would be necessary to further constrain the generator to adhere more to its input or to transform the labels in a similar fashion.

Using GANs to generate additional training data for classification tasks can be a worthy aim. For example, in [138] realistic images of cylinder heads are generated from coarse labels. The generated images contain defects close to the regions defined in the labels. Here, the parts are classified as a whole, so the generator does not need to maintain the precision of its input. In contrast, we need to precisely segment individual defects within a part. Consequently, training a GAN to refine the training data for a precise labeling task is a hard endeavor with an uncertain outcome. The GAN tends to alter the shape information so that the data does not match the labels anymore. Using too restrictive *consistency constraints* inhibits the possibilities of the GAN to add more realism to the

data. Moreover, the generated CT scans are not explainable to domain experts. Hence, we abandoned this approach.

3.2.3. Simulated Data

The last approach for closing the gap between synthetic and real data is to model the physical conditions of the real world more accurately. *More realistic simulations*, which even domain experts have a hard time recognizing, should improve the predictions of our model, too. Basically, there are two ways to achieve more realistic simulations: (i) we can build upon existing real-world CT data and render artificial defects in real-world CT scans. If we further have the projection data, we can superimpose the defect in the projections and later reconstruct the CT scan from the manipulated projections. It is conceivable to use more sophisticated techniques of superimposition, which respect the laws of physics, altering the attenuation instead of the gray-values directly [139]. (ii) we can simulate the complete data, including the shape of the cast part, all the defects, and all the artifacts which occur in the real world [140–142]. Using approach (i) we need to consider that the CT scan in which we place our artificial defects might already show real defects, as cast aluminum parts usually are never free of flaws. These need to be considered in the labels, too. Furthermore, a defect has not only an influence at the position where the defect actually is but on the entire CT scan. These effects would not be considered. Approach (ii) yields by far the most realistic synthetic data which is able to even fool domain experts. We are able to simulate all the effects a defect has throughout the entire CT scan and, in contrast to GANs, which are limited by their receptive field, the simulation also considers global artifacts like ring or streaking artifacts. On top of that, realistic simulations already have been successfully used to carry out experiments regarding the evaluation of measurement uncertainty in CT scans [143, 144]. Therefore, we decide to go for this approach, even though it entails a considerable computational effort.

Using this realistically simulated data with precise labels for training, we obtain *crisp edges* and are able to detect even *small defects* (see Figure 3.11). Another benefit is that we use data which is explainable to domain experts. The process to obtain realistic training data is shown in Section 3.3. Of course, we cannot find each tiny defect. An instance needs to have at least three voxels in diameter in order for the model to properly distinguish between defect and noise. More detailed results on that follow in the next chapters.

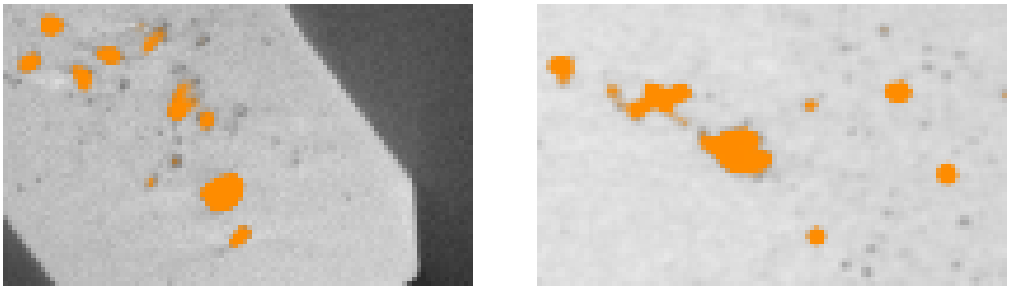


Figure 3.11.: The model which is trained with precisely labeled and realistically simulated CT scans yields a crisp segmentation mask and manages to detect even small defects. However, instances which are smaller than three voxels in diameter remain undetected. More detailed results are shown in Chapter 5 and Chapter 6. The orange overlay shows the prediction of the trained model.

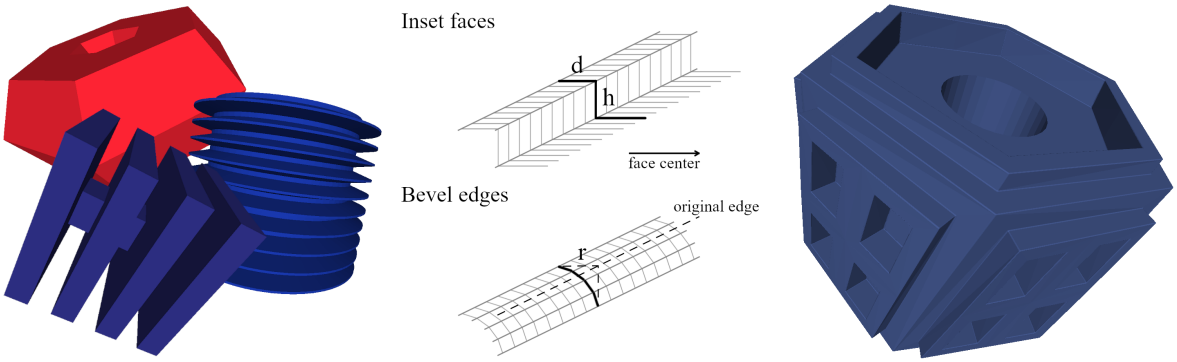
3.3. A Fully Automated Simulation Pipeline

Simulating the necessary amount of data manually by shaping and placing defects in computer-aided design (CAD) files, setting up the virtual scanner for every scan and operating the reconstruction and ground truth generation is not feasible at all. Therefore, we have to automate each of these steps: First, we procedurally generate the meshes of materials and defects; then, we script the simulation of realistic projections; and, finally, we automatically reconstruct the CT scans and compute the ground truth. With this pipeline at hand, we are able to obtain the necessary amount of precisely labeled three-dimensional volumetric data for training.

3.3.1. Procedural Modeling of Defective Castings

The first step is to automate the generation of *virtual cast parts*. We decided to generate their shapes instead of using real CAD models. This avoids both compliance issues and unintended bias towards specific shapes of real cast parts. However, we need to be able to generate any number of different shapes to cover the essential aspects of real cast parts. Hence, we design our virtual cast parts in a way that they cover randomly chosen aspects of real parts, for example, inlets, thin webs, or drill holes. In the development of computer games we encounter similar requirements, for example, for the generation of trees. They should all look alike, but neither should look like the other for a more realistic experience. Thus, they are often created by an algorithm which randomly "grows" them segment by segment [145, 146]. This is called "*procedural modeling*". Another application is, for example, the creation of buildings from combining elementary components like walls, roofs, windows, and doors. They are combined following certain rules which define where doors and windows can be placed and how many components are allowed [146, 147]. With such methods it is possible to generate entire worlds. Certainly, a realistic appearance requires sophisticated and well chosen rules. [146]. To automatically generate a wide range of virtual cast parts, we utilize some fundamental techniques of procedural modeling.

Our goal is to generate our virtual cast parts as objects of different shapes with varying penetration lengths. The occurring partial volume effects render dark streaks in the CT scans and make them more challenging. We, therefore, define a set of *basic elements* $\mathbb{E}$ and a set of *operations* $\mathcal{O}$ to combine and refine them. Then we can randomly chose from $\mathbb{E}$ and $\mathcal{O}$ to create any number of differently shaped and more complex elements. $\mathbb{E}$ includes a cube, a prism, a frustum, a regular polyhedron, and a toroidal polyhedron. The cube is defined by its edge length a. The frustum is defined by the radius r_M of its basis, the radius r_m of its top, its height h and the number of sides n. Basically, the prism is a frustum with $r_M = r_m$. The regular polyhedron is defined by its radius r and the number of sides n. The toroidal polyedron is defined by its major radius r_M, its minor r_m, the number of segments n_M along r_M and the number of segments n_m along r_m. Furthermore, we create two more sophisticated elements: a fin structure and screw threads (see Figure 3.12a). They are defined by their number of edges n, their height h, their major radius r_M, their minor radius r_m, and their alternation distance d of wide and narrow parts. These elements are grown by face extrusion from a prismatic basis, altering their diameter between r_M and r_m. For the fin structure we chose $n = 4$ with $r_m << r_M$. For the screw threads we have $n = 64$ and a much smaller difference between r_m and r_M. Additionally, all elements of $\mathbb{E}$ further have a location p and an orientation ε.



(a) The fin structure and a screw thread. (b) The inset and bevel operations. (c) Example of a virtual cast part.

Figure 3.12.: Our virtual cast parts are generated procedurally by randomly choosing from a set of basic elements and set of operations to combine and refine the basic elements.

The set of operations $\mathcal{O}$ is defined as follows:

Boolean operations These operations are defined in terms of the difference $\setminus$, the union $\cup$, and the intersection $\cap$ of the volumes which the meshes enclose. These operations require the meshes to be "*watertight*", i. e. a 2-*manifold* without boundary. This is necessary to properly define the intersection points of the two meshes. While these operations are mathematically well defined, the actual implementations most probably suffer from floating point imprecision which might break the 2-manifold of the resulting mesh [148].

Subdivide surface As the low-polygonal nature of our basic elements limits their variety, we define a surface subdivision operator $\mathcal{S}$ to split large faces into multiple smaller faces by placing n equally distributed cuts along each edge.

Inset faces The inset operation $\mathcal{I}$ either produces an indent or an elevation of the faces of the mesh. It duplicates the corner vertices of a face, and moves them towards the center of the face by d. Then, these vertices are duplicated again and moved into the volume or out of it by h (see Figure 3.12b). To maintain the 2-manifold, the faces are re-wired after the duplication of the vertices. We define d and h as a fraction of the edge length of the face.

Bevel edges When a cast aluminum part comes out of its mold and before it is machined, its edges tend to be rounded off. Therefore, we use the bevel operation $\mathcal{B}$ which introduces n additional faces at the edges of a given mesh and distributes their corner vertices in a given radius r over the edge to round it (see Figure 3.12b).

Simple deformations For further variations the deformation operations "twist" $\mathcal{T}$, "bend" $\mathcal{D}$, and "roughen" $\mathcal{R}$ manipulate the meshes. $\mathcal{T}$ rotates the vertices of one end against the vertices of the other end by α around a given axis. $\mathcal{D}$ compresses the vertices of one side of the volume and expands the vertices on the other side by f on a given axis. $\mathcal{R}$ works on the surface of the mesh: it subdivides large faces to a fine grid and randomly moves the vertices a bit along their normal. With that we can simulate the rough surface of unmachined parts.

With the formula of Equation (3.1), for instance, we generate the virtual die cast M which is shown in Figure 3.12c. While arbitrary random combinations are conceivable, this could lead to severe

problems: self-intersections and holes which both yield non-manifold meshes. This means, we have to rule out all malicious combination. Instead of defining complex rules how to chose from $\mathbb{E}$ and $\mathcal{O}$ and how to configure the elements and operations, we simply select a set of 25 different combinations and additionally define two models as validation data. As we have scans of different configurations for each part we consider this amount of data as sufficient. If the need for more data emerges, we can come back to this step at any time.

$$M = \mathcal{T}\left(\mathcal{B}\left(\mathcal{I}\left(\mathcal{I}\left(\mathcal{S}\left(\mathbb{E}_{\text{prism}}\left(r_a, h_a, n_a\right)\right)\right)\right)\right) \setminus \mathbb{E}_{\text{prism}}\left(r_b, h_b, n_b\right)\right) \tag{3.1}$$

All our virtual cast parts are designed to have about the same size. We create them to have a penetration length of about 80 mm. Therefore, for the example given in Equation (3.1), we chose $r_a = 40\,\text{mm}$, $h_a = 45\,\text{mm}$, and $n_a = 6$ and $r_b = 10\,\text{mm}$, $h_b > h_a$, and $n_b = 64$, respectively. Both $\mathbb{E}_{\text{prism}}$ are positioned at $p = (0, 0, 0)$ with an upright orientation $\varepsilon = (0, 0, 0)$. For simplicity, we omit the parameters of the modifiers.

Having our virtual cast parts in place, we need to fill them with flaws and defects—our actual target class. To get as close to reality as possible, we need to model different types of flaws: gas pores, shrinkage cavities, solidification cracks, structural loosening, and inclusions of foreign material. Furthermore, each category requires a different placement strategy. For the creation of the defect meshes we adhere to current guidelines and standards which deal with the classification of different defect types [1, 9, 149]. During our research, we also examined hundreds of examples of real defects and extracted some polygon meshes of the surface of those defects to gain more information about their shape. We could simply place these extracted defect meshes in our virtual cast parts over and over again, varying orientation and size. Instead, we again do not use the models of real defects to avoid an unintended bias towards specific shapes but come up with the following rules to procedurally model an arbitrary amount of unique defects.

Gas pores A roundish, compact appearance characterizes this type of defect. It seems logical to start off with a sphere and step by step add further spheres using Boolean operations. However, this increases the risk to violate the 2-manifold of the defect mesh, e. g. by creating holes in the mesh or by introducing self-intruding or mis-oriented faces due to possible floating point imprecision in the implementation of the Boolean operations. Hence, we start with a cube instead and extrude its side faces to alter its appearance. Then, we smooth the surface, which introduces new vertices. To increase the variety of the gas pores we now move individual vertices along their normal. Finally, we need to restore the roundish character of the gas pores by a second smoothing (see Figure 3.13).

Shrinkage cavities Those cavities have a rugged surface with spikes and sharp edges and corners. Again we start off with a cube. This time, however, we extrude its sides iteratively, as we do not need to maintain a roundish shape. After smoothing the surface, we randomly move the individual vertices, too. For the rugged surface we further select a few vertices which are farther dragged out. Finally, the surface is smoothed again (see Figure 3.14).

Solidification cracks With their sharp corners they tend to wind through the material. Here, we start with a cuboid and define a special rule for the face extrusion: A flat element like the starting cuboid can only be extruded at its small sides by a distance smaller or equal the width of the small side. This new part then is a corner. At corners only the long faces can be extruded

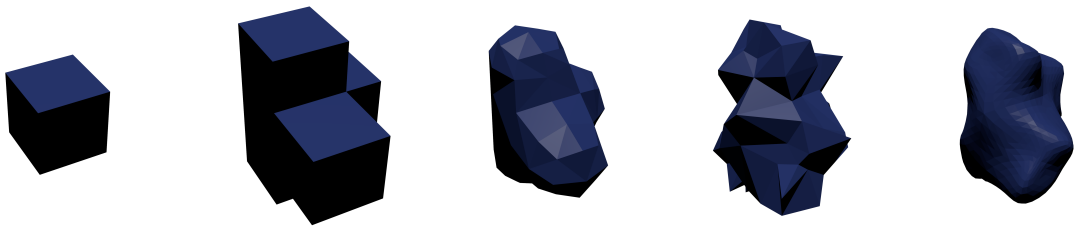


Figure 3.13.: Procedural modeling of gas pores: We start with a cube and randomly extrude some of its faces to create the basic shape of our bubble. Then, we do a first smoothing of the surface. This introduces new vertices to our bubble. To make it more interesting and unique, we randomly move the individual vertices along their normal. Finally, we smooth out the surface to obtain a roundish bubble again.

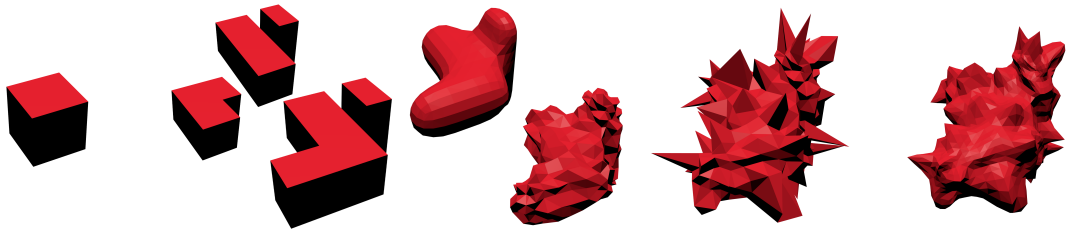


Figure 3.14.: Procedural modeling of shrinkage cavities: Again, we start with a cube. This time we iteratively extrude random faces to create the basic shape. Then, we do a first smoothing of the surface and randomly move individual vertices along their normal. To attain the rugged surface with spikes and sharp edges, we pick a few vertices again and increase their translation. Finally, we smooth the surface again to obtain the shrinkage cavity.

to form either a second corner part or a new flat element. For now we only have rectangular bends in the cracks, but it is conceivable to introduce arbitrary angles by converting the corner parts to trapezoids. Then, we again subdivide the surface and randomly move the vertices along their normal. Optionally, we twist or bend each instance to get more variety into our cracks (see Figure 3.15).

Structural loosening Modeling a loosening of the structure is quite a hard problem: usually it is an accumulation of small pores or a sponge-like pore network, in which the individual pores cannot be resolved individually by the CT scan. These pores, thus, appear as an area of lower material density. Simply placing a myriad of tiny gas pores to simulate a structural loosening

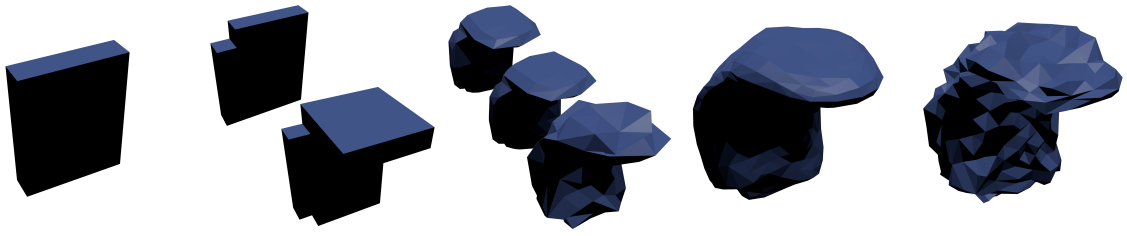


Figure 3.15.: Procedural modeling of solidification cracks: Here, we start with a cuboid. During the iterative extrusion of faces we distinguish to cases: (i) the creation of corners and (ii) the creation of sides. Next, we subdivide the surface, randomly move the vertices along their normal, and add a little twist to obtain the small cracks which wind themselves through the cast aluminum part.

tremendously increases the computational effort. Moreover, we need to take care of the 2-manifold when merging the meshes of the individual pores. Growing a large blob throughout the virtual cast part produces less computational effort, but it introduces a quite abrupt change in material density, while a real structural loosening has a more smooth transition. To model the smooth transitions we try star-like spikes around the grown structural loosening so that the density slightly decreases. However, this star pattern appears in the CT scan—especially if the spikes are orthogonal to the projection plane. Therefore, we abandoned this approach and go with the simple blobs for now. We grow the instances of structural loosening similarly to the shrinkage cavities: We only create them larger and leave out the creation of spikes. The growing process is guided so that the structural loosening avoids the boundaries of the virtual cast part.

Large splits Another type of defects are large splits which meander through out a large part of the material and branch often. These defects often occur due to material fatigue [150]. To model these splits we start off with a two-dimensional plane of $n \times m$ vertices. To give the split its meandering appearance we use the "twist" and "bend" operations which we defined for the creation of our virtual cast parts. Then, we randomly move the vertices along their normal to get a more rugged surface. Finally, we extrude all faces at once by a small distance to give the split a volume. Instead of branching, we simply create two overlapping splits, which we combine using the union operation.

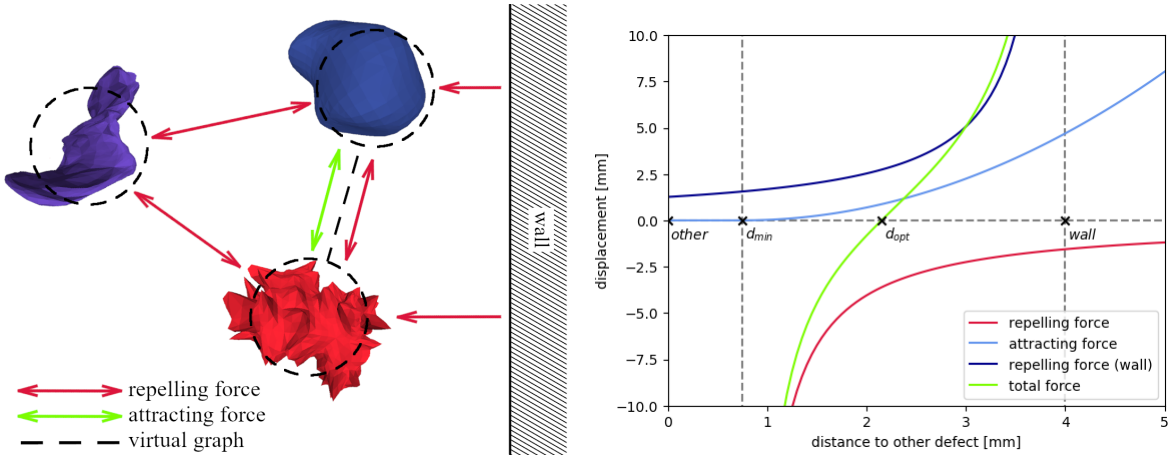
Inclusions of foreign material They occur, for example, when the admixtures in an aluminum alloy react with the oxygen in the mold [8]. This way, small portions of foreign material can be formed, which have different material properties than the alloy itself, e. g. spinel, which is harder than the surrounding aluminum alloy. Thus, we model them as small, simple blobs and assign them a proper material of higher density later on.

Now that we can model the different defect types, we need to place them in our virtual cast parts. The different defect types have different requirements regarding their placement. For some it is okay to just place them randomly, others require a more sophisticated strategy to make the final simulation look more realistic.

Random positioning The most simple positioning strategy is to choose random points inside a given mesh. While this method is pretty fast, it offers no control about where the defects are placed—except for being placed somewhere inside a given mesh, even close to the surface.

Surface distribution The same holds for the surface distribution strategy. Here, we chose random points, too, but this time we explicitly sample on the surface of the mesh, i. e. our material boundary. This allows us to create a separate class which only contains those defects which penetrate the surface of our virtual cast part. This is necessary as our machine learning defect detection algorithms tend to yield false positive predictions at the material boundary, for example, in screw threads, when only training "defect vs. defect-free". Using a separate class for surface defects, we are able to shift the false positives to this separate class so that we get more pure results for the inside of a cast part. (Training with screw threads in the training set further improves the results, see Section 4.3.4.)

Cluster forming When exploring the different types of defects, we often came across agglomerations of defects. Therefore, we like to model these defect clusters explicitly. Instead of using



(a) We assign a node to every defect and randomly draw connections between them to obtain our virtual graph. Between every node there is a repelling force which drags them apart and between connected nodes there is an additional attracting force. Further a repelling force from the wall keeps the defects inside the virtual cast part.

(b) The displacement (the strength of the force) as a function of the distance from another defect. Additionally, we added the repelling force induced by a wall. Positive displacement values push our defect closer to the other defect, negative forces pull them farther apart. In the optimum d_{opt}, repelling and attracting forces cancel each other out.

Figure 3.16.: We use a force-directed graph-layouting algorithm to distribute our defects so that they form clusters.

time-consuming and expensive casting simulations, we make a little abstraction and approximate this behavior, borrowing a method from the visualization of graph structures: We use a modified version of the *force-directed graph-layouting algorithm*, known as the Fruchterman-Reingold algorithm [151]. The goal of visualizing a graph is that connected nodes stay close together while other nodes move apart—and of course, the nodes should not overlap. The Fruchterman-Reingold algorithm achieves this goal by defining a *repelling force* between each and every node and additionally assigning an *attracting force* to connected nodes. For our purpose we need to define a virtual graph referencing our defects. Each defect represents a node of this graph. Then, we randomly draw connections from one node to other nodes which are close by. This forces the formation of defect clusters. Furthermore, it is necessary to add repelling forces between each defect and the material boundary which is defined by our virtual cast part (in fact, it could be any mesh). Figure 3.16a gives a notion of how the algorithm works.

In the original algorithm the repelling force is defined as $f_r = -k^2/d$ and the attracting force as $f_a = d^2/k$, where k is an optimal distance which is defined in terms of the available space, the number of nodes, and an experimental factor [151]. As we employ a strict boundary instead of a vague "available space" we define our optimal distance in terms of the radii of the involved nodes/defects u and v as $d_{\text{opt}} = c_{\text{opt}}\,(r_u + r_v)$. The minimal distance is defined analogous as $d_{\text{min}} = c_{\text{min}}\,(r_u + r_v)$. The factors c_{opt} and c_{min} control the density of the clusters with the constraint $c_{\text{min}} < c_{\text{opt}}$. The actual distance d_{actual} is defined as the Euclidean distance of the center of the nodes. With these distances we define our forces as shown in Equations (3.2), (3.3), and (3.4). Figure 3.16b shows for two defects and a wall how the forces manage to move the defect in its (locally) optimal position.

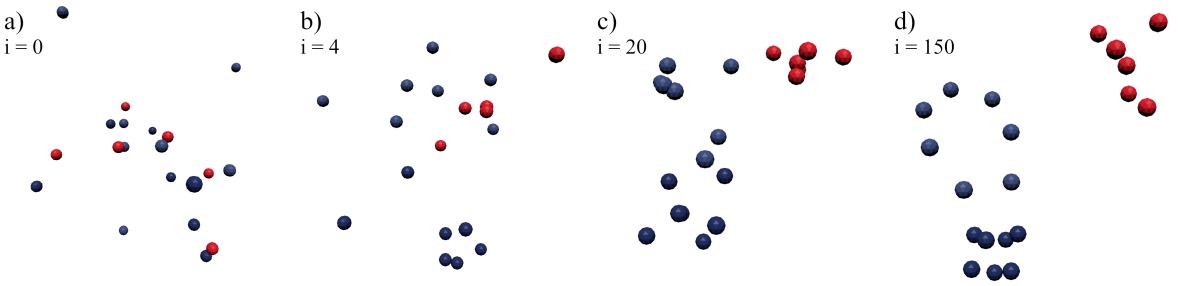


Figure 3.17.: Defect placement: The Figures a – d show the process of forming three clusters from a random distribution at different iterations i. We form a compact cluster of densely connected nodes (dark blue), a ring (light blue) where each node is connected to two neighboring nodes, and a star-shaped cluster (red) where each node is connected to a central node. Furthermore, the ring and the densely connected cluster are connected by two nodes so that these clusters stay close together.

$$f_r = \frac{(d_{\text{opt}} - d_{\text{min}})^2}{d_{\text{actual}} - d_{\text{min}}} \tag{3.2}$$

$$f_a = \frac{(d_{\text{actual}} - d_{\text{min}})^2}{d_{\text{opt}} - d_{\text{min}}} \tag{3.3}$$

$$f_{r_{\text{wall}}} = \begin{cases} \frac{(d_{\text{opt}} - d_{\text{min}})^2}{d_{\text{actual}} - d_{\text{min}}}, & \text{if is inside} \\ d_{\text{actual}} + d_{\text{min}}, & \text{otherwise} \end{cases} \tag{3.4}$$

The force-directed graph-layouting is an iterative algorithm. In each iteration we compute the strength and direction of all f_r, f_a and $f_{r_{\text{wall}}}$ for each node and add them up. For simplicity we do not calculate any repelling forces if $d_{\text{actual}} > d_{\text{influence}}$, where $d_{\text{influence}}$ is a given distance beyond which we say that the defects are independent of another (or the wall). This gives us the desired displacement for all defects, which can be quite huge, e. g. when one defect u is close to a group of other defects, they all put strong repelling forces upon u, which all point in about the same direction. Therefore, the displacement Δ_i of a node u in iteration i is bounded by the current "temperature" t_i of the system: $p_{i+1} = p_i + \Delta_i / |\Delta_i| \cdot \min(t_i, |\Delta_i|)$, where p_i is the position of a node (or defect) in iteration i. After updating the position of all nodes we "cool down" t_i by a cooling factor $c_{\text{cooling}} = 0.95$. The initial value of t depends on the size of our virtual cast part and is about 10 % of its diameter. The iteration stops after we (a) reach a given maximum number of iterations, or (b) the nodes do not move anymore. Figure 3.17 coarsely outlines the process of defect placement.

In each of our virtual cast parts we try to place about 650 inner defects using the cluster forming strategy. For the inner defects we randomly choose from gas pores, shrinkage cavities, and small cracks. Up to now the large splits are only part of a proof of concept dealing with fatigue cracks occurring, for instance, in nickel alloys (see Section 6.2.4) and, hence, are not included in the data set. We remove instances which either penetrate the material boundary defined by our virtual cast part or intersect with surface defects. Furthermore, we rule out instances which are not a 2-manifold anymore. Therefore, the exact number of defects can only be determined afterwards. Usually, we end up with about 610 defects per virtual cast part. The diameter d of an inner defect is chosen from a normal distribution $\mathcal{N}\left(\mu = 0.7\,\text{mm}, \sigma^2 = 0.6\,\text{mm}^2\right)$ and limited to the range $[0.2\,\text{mm}; 3.0\,\text{mm}]$. After creating the actual shape of a defect, it is scaled to fit in a bounding sphere with diameter s. This might cause the actual defect to have a far smaller volume than its original bounding sphere. Moreover, we place place about 50 defects with $d = \mathcal{N}\left(\mu = 2.0\,\text{mm}, \sigma^2 = 1.0\,\text{mm}^2\right)$ capped at $[1.0\,\text{mm}; 3.0\,\text{mm}]$

using the surface distribution strategy. To consider structural loosening in our training set, we grow 40 instances throughout our virtual cast parts originating at randomly chosen seed points. Additionally, we grow another 10 smaller instances inside the other instances of structural loosening. This adds an additional variation to the material density and such adds a little more realism to the data. Finally, we randomly place 75 inclusions of foreign material in the virtual cast part. Their size is drawn from an uniform distribution $\mathcal{U}$ (0.3 mm, 0.8 mm). To ensure that the inclusions do not float in the void of another defect, we first subtract all defects from the virtual cast part and use the resulting mesh as input for the random positioning.

Currently, we are using Blender (Blender Foundation) to automate the creation of our virtual cast parts and the different defect meshes. However, as we only use simple mesh operations, switching the underlying framework to, for example, a game engine should be easily possible if needed.

For our training set of cast aluminum parts we currently restrict our algorithms to the detection of gas pores, shrinkage cavities, small cracks, occurrences of structural loosening, and inclusions of foreign materials. Although the deep learning method which is trained with this data set of cast aluminum parts also works quite well, in other domains like additive manufacturing (see Section 6.2.3) we have to come back to the mesh creation to further improve the results. To really make the transition from cast aluminum to additive manufacturing, we have to consider new defect types which are unique to additive manufacturing like inclusions of remaining powder or flatter gas bubbles (see Section 6.2.3).

3.3.2. Realistic Simulation of Projections

Realistic simulations involve a considerable computational effort. To realistically simulate the projections for the CT scans, we use a *ray-casting* method [141]. For each pixel on the detector, a number of rays is cast throughout the scene starting at the X-ray source. When passing through the scene the poly-chromatic spectrum each ray carries is attenuated according to the absorption coefficients of the material which is penetrated. For this ray-based simulation of realistic projections we use aRTist (Federal Institute For Materials Research and Testing, Germany) [141], which is widely accepted among the domain experts. As the application by default produces single projections, we extend it with a plug-in module which allows us to automate the acquisition of a CT scan, i. e. automatically rotating the virtual cast part and simulating a projection at defined angles. The plug-in module further automates the loading of the meshes of our virtual cast parts and setting up the CT system to pre-defined configurations.

Before we setup the CT system, we configure the scene by composing our virtual cast part from the individual meshes which we created in the previous step. In aRTist the meshes override each other when it comes to the calculation of the attenuation: For overlapping meshes, only the mesh that comes last in the scene tree is considered. Therefore, we need to place the boundaries of our virtual cast part first, followed by the structural loosening meshes and the inclusions. Lastly, we place the defect meshes. Then, we assign a material to each mesh, which defines its absorption coefficients. In aluminum casting pure aluminum is used rarely. To alter the strength, hardness, formability, or other properties of the final product, an alloy with admixtures like copper, manganese, or silicon is used. These admixtures have an influence on the absorption coefficients of the material as shown in Section 2.1.3. As we like to have a realistic simulation and to train a classifier with a broad application

spectrum, we choose an universal alloy which is widely used in automotive and aerospace industries. It is defined by the standard EN AW 2014 [36] and has admixtures of copper, iron, silicon, and a few others. The material has a density of 2.80 g cm^{-3}. In fact, the simulation pipeline would allow us to alter the material in every scan. For reasons of simplicity and because the resulting CT scans of different aluminum alloys look quite similar from an image processing point of view, we only choose a single material. The instances of structural loosening are assigned the same material as the virtual cast part—only at lower density. We randomly assign them a density of 95 %, 90 %, 85 %, or 80 % of the density of the material we use for the virtual cast part, making sure that overlapping instances of structural loosening are assigned different densities. The inclusions in our training set are formations of spinel ($MgAl_2O_4$) with a density of 3.60 g cm^{-3}.

Now that we have our virtual cast part composed and ready, we need to configure the virtual CT system. However, the parameter space of a CT system is huge and offers a vast amount of possible configurations (as we show in Section 2.1.1). Again, from an image processing point of view many of the possible combinations of parameters yield quite similar gray-value images and only slightly affect the detection rates of defect detection algorithms. We assume that an operator of a CT system knows how to tune the individual parameters to make a CT scan in which the smallest defects which are to be found can at least barely be resolved. This allows us to simplify the parameter space for the realistically simulated CT scans of our virtual cast parts keeping most of the parameters fixed. Then, we only need to vary those parameters which most significantly cause the image artifacts which impede an automated defect detection. Thus, we set up our system, using a 225 kV source at 1 mA. With that we can maintain a focal spot size of 225 µm × 225 µm. As we only need to scan parts with an edge length of about 80 mm we choose a relatively small detector with 1000 × 1000 pixels and a pixel size of 200 µm × 200 µm. The distance from the source to the detector is 1000 mm and the distance from the source to the object is 450 mm. This gives us a magnification of about 1.82 and we, therefore, end up with a voxel size of 110 µm × 110 µm × 110 µm. With our smallest defects having a diameter of 200 µm they barely have a diameter of two voxels and, hence, are the most challenging instances in our data set. To explicitly simulate impeding artifacts in our data, we (i) change the exposure time to vary the amount of image noise, we (ii) change the thickness of the filter of the source to vary the influence of beam hardening, and we (iii) change the quality of the detector calibration to vary the strength of ring artifacts in the reconstructed CT scan. We so reduce the parameter space of a CT system to a three-dimensional sub-space, the "*artifact space*", which is described as follows:

Noise The noise in CT scans mostly arises from the quantum nature of X-rays, i. e. the variations in the number of photons reaching the detector (quantum noise) [40, 152]. This type of noise can be modeled using a Poisson distribution. Other sources of noise arise from electrical noise of the detector, which is of particular significance in under-exposed regions, and quantization errors due to limits in the dynamic range of the sensor [33, 34]. While the quantum noise can be mitigated by longer exposure times and averaging of multiple projections, the other types of noise require more sophisticated hardware. In aRTist the image noise is added after doing a "perfect" simulation depending on the gray-value in the projection using a Poisson distribution [40, 140]. To span up a range of different noise-levels we use a virtual *exposure time* of (i) 1 s, (ii) 0.5 s, and (iii) 0.25 s. This yields CT scans with a CNR of about 12, 6, and 3, respectively.

Beam hardening Actually, this category should be named "cupping artifacts". However, as we

tackle only the portion of cupping artifacts that arises from beam hardening, we decide to name this dimension of the artifact space "beam hardening" for clarity. Instead of using a computational intensive Monte Carlo simulation which describes the stochastic absorption and scattering processes in detail for every projection, aRTist uses a ray-casting approximation based on the attenuation law taking into account the penetration length and the energy spectrum [140]. To span up a range of different levels of cupping artifacts, we vary the *prefilter* of the source which reduces the amount of low-energy photons. We use (i) no filter at all, (ii) 0.5 mm of copper, and (iii) 1.0 mm of copper.

Ring artifacts This type of artifact is not simulated in aRTist as it is rendered into the data while doing the reconstruction of the CT scan. In real world CT scans ring artifacts arise from an inadequately calibrated detector [153, 154]. Therefore, for our simulations, we artificially distort the calibration files which are used during reconstruction. For now, we only change the noise-level of the bright image (I_B). To create more severe ring artifacts, it is conceivable to further manipulate the dark image (I_D), or to even manipulate the projections to introduce bad pixels. To span up a range of different levels of ring artifacts, we use *bright images* with (i) 20 %, (ii) 60 %, and (iii) 100 % of the noise of the actual projections when reconstructing the CT scan.

Instead of random sampling, we simulate all three gradations of each artifact type for each of our virtual cast parts. This allows us to draw conclusions about the effect of individual artifacts on the automated detection of defects (see Section 5.3.2). Because not all artifacts can be mitigated by varying a parameter of the CT system, we do not vary the severity of those artifacts but set them to a realistic level and simulate them in the same fashion throughout all CT scans.

Scatter. Scatter is a significant influence factor impeding the detection of defects. Unfortunately, it is pretty hard to mitigate the effect of scattering with proper collimation [33, 155]. Therefore, we simulate the same amount of scattering in all our CT scans without using any collimation. For the simulation of scatter, aRTist uses a Monte Carlo method which traces a given number of photons (in our case $2 \cdot 10^7$) throughout the scene and simulates Compton scattering along that path [141]. The photons are traced until they hit the detector or leave the frustum of the scanner. For all pixels of the detector to be hit by at least a single photon, we would need to simulate many more photons. Thus, aRTist interpolates between the pixels that contain a piece of information from the simulation. As this method is highly computationally expensive, aRTist offers a relaxation: The precise scatter simulation is done only every x projections and stored in a separate scatter image which is then added to every projection. However, aRTist only considers the scattering that happens between source and detector and neglects the scatter introduced by the surrounding walls of the chamber of the CT system. Unfortunately, aRTist also neglects the internal scattering in the detector, which, nonetheless, has a huge influence on the outcome. The internal scattering has an opposing effect to the scattering in the object. The scattering in the object brightens the outer regions close to the material boundary, the reconstructed object appears smaller. The internal scattering of the detector brightens the regions of the detector which are covered by the object, the reconstructed object appears to be larger [41].

Partial Volume Effect. Typically, parts should be placed in the CT system in a way which avoids widely varying penetration lengths and edges which are aligned orthogonal to the detector [33, 153, 154]. The first is to ease the effects of beam hardening; the latter to avoid impediments from partial volume effects. Despite that, we do not tilt our virtual cast parts, which would mitigate the artifacts arising from the partial volume effect, because not all geometries allow a proper placement. This way we obtain more artifacts in our CT scans, but this prepares our algorithms for CT scans in which these effects cannot be mitigated easily.

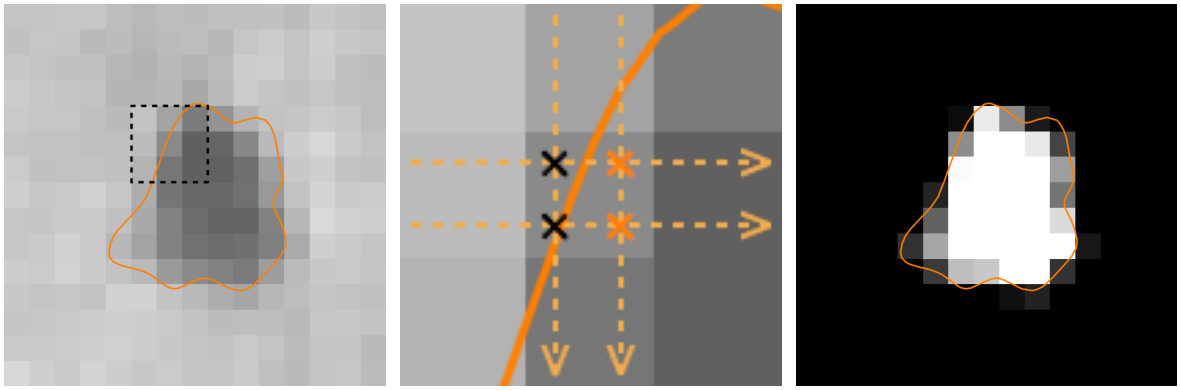
Aliasing Artifacts. We acquire full 360° scans but only simulate 720 projections, i. e. one projection every 0.5°. This leads to an under-sampling and, thus, introduces minor aliasing artifacts [33, 153, 154]. These artifacts are especially visible in the outer regions of the CT scan because, here, the gap introduced by the angle step is wider than the voxel size. This presents another challenge for our defect detection algorithms.

Usually, the parts are placed in some kind of mount. We do not model these mounts yet. Most of the times, they are made of plastics which in contrast to our aluminum parts are easy to penetrate. When switching to other materials the mounts can be critical, but for now their effect on the scan is negligible. Neither do we change the appearance of the focal spot yet. We use a perfect Gaussian distribution. However, it is possible that the focal spot has a more elliptical shape and probably changes throughout the scan as the source heats up [156, 157]. This happens especially when using long exposure times. This is denoted as focal drift and leads to further inaccuracies in the reconstruction of the CT scans. The photons which arrive at the detector are not (yet) counted directly. There is a scintillating layer on the detector which converts the X-ray photons into visible light, before the detector responds to that signal [33]. The "translated" photons can, therefore, scatter within the detector. Moreover, this "translation" is not a binary effect; the scintillating layer glows for some time, after being hit by a X-ray photon. This means, if the time between two projections is too short there is an afterglow of the previous projection in the current one [158]. Both effects can have an impact on the CT scan but are not yet part of the simulation.

To automate the simulation of CT scans we need to extend aRTist by writing a plug-in. The plug-in scans a given directory for model definitions, i. e. the virtual cast parts from the previous step. Then, it automatically loads these virtual cast parts including their defects one after another into the scene. Given our set of configurations, the plug-in configures the parameters of the virtual CT system and orchestrates the CT scan, i. e. making the realistic projections and rotating the virtual cast part. Currently, the plug-in is restricted to cone beam CT, but different imaging techniques like helix CT can be implemented analogously. Finally, with the help of the VG PROJECT SDK (Volume Graphics GmbH), our plug-in module creates the project files which contain the location of our projections on the hard drive and the instructions how to reconstruct the CT scan. These project files form the entry point for the next step of our fully automated simulation pipeline.

3.3.3. Bring Your Own Ground Truth

For the reconstruction of the CT scan from its individual projections we use the CT reconstruction module of VGinLINE (Volume Graphics GmbH). This connects seamlessly into our pipeline: It



(a) The defect mesh is precisely aligned with the gray-value grid of the CT scan.

(b) Via ray-based sampling we determine the fraction of each voxel which lies within the defect mesh.

(c) The computed ground truth is stored as an 8 bit mask.

Figure 3.18.: Exemplary computation of the precise ground truth, which we need for a proper training. The orange outline shows the defect mesh which was used to simulate the underlying gray-value grids. The dashed outline in Figure (a) corresponds to the patch which is shown in more detail in Figure (b).

looks for new project files which we create in the previous step and processes the reconstruction information they contain. The last step is to compute the ground truth.

Knowing exactly how our virtual cast part and its defects are placed in our virtual CT system allows us to precisely align the mesh data of the defects with the reconstructed voxel grid (see Figure 3.18a). Thus, we can determine for each voxel the fraction which lies inside the given mesh, i. e. the fraction which is part of a defect. For an easier computation we approximate this value by placing a regular grid of $n \times n \times n$ sample points in each voxel and determine for each sample point whether it is *inside* or *outside* of the mesh. The sample points lie inside the voxel not at its boundaries. So we avoid sampling two (or even eight times) at the exact same position (see Figure 3.18b).

Actually, we cast $n \times n$ rays through each row of voxels storing their state *inside* or *outside*. The initial state is *outside*. Then, we determine the intersection points of each ray with the given mesh. Each intersection changes the state. This approach significantly speeds up the ground truth computation. In exchange, it introduces a new source of errors: if we miss an intersection even though the mesh is "watertight", for example, due to the limited accuracy of single precision floating point numbers, we end up with long streaks which extend over the entire volume. To mitigate these artifacts we scan the volume three times; once along every axis. Then, we combine the three masks using a majority vote. Figure 3.18b shows a simplified two-dimensional version of this approach with $n = 2$.

For our ground truth we choose $n = 8$, so we get 512 sample points per voxel. We store our ground truth as 8 bit unsigned integer, which holds up to 256 different values. This means we have 1 bit redundancy. Figure 3.18c shows an example of how the voxel-precise ground truth looks like. For training, however, we often require a binary mask. Depending on the threshold t we use to binarize the mask we are able to steer the sensitivity of the trained classifier. For example, with $t = 16$, i. e. only 6.25 % of the voxel belong to a defect, the trained classifier will recognize way more voxel as "defective" than a classifier trained with $t = 128$ (50 %).

To obtain a more precise ground truth, we introduce a *sub-sampling factor* which allows us to get

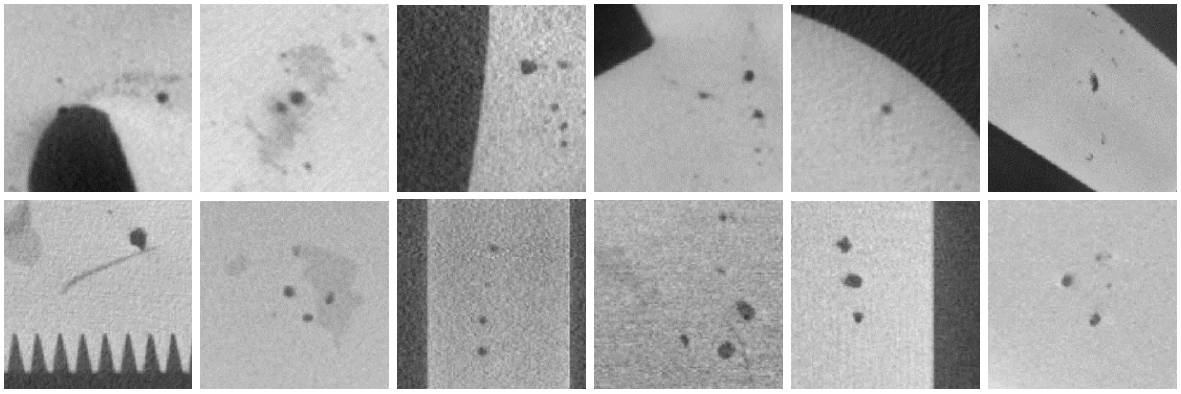


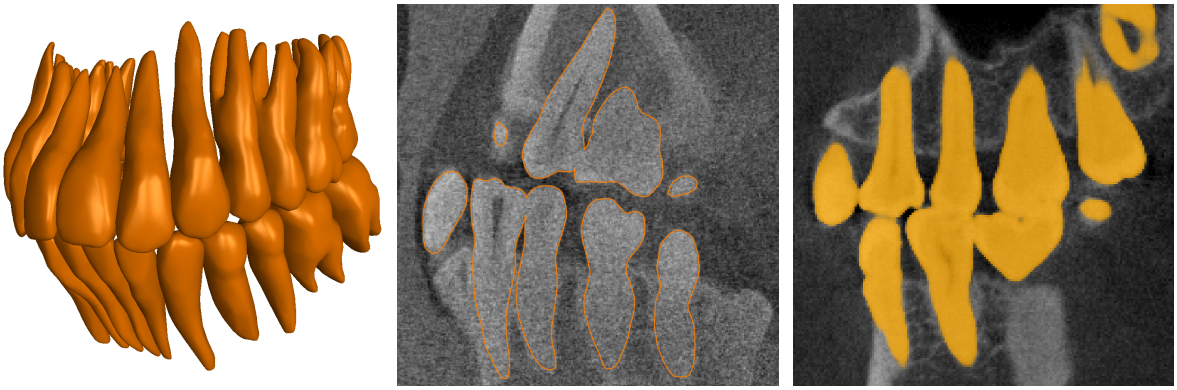
Figure 3.19.: A comparison of patches taken from real CT scans to patches of our realistically simulated training set. The top row shows the real images, the bottom row the simulated images. A closer look reveals the most significant drawback of the simulations: The simulated CT scans lack the fine details of real CT data. For example, we miss out the small fluctuations in the density of the material. Thus, the simulated CT scans have a more homogeneous structure.

eight, 27 or even 64 ground truth voxels per data input voxel. This way, we can compute an arbitrarily precise ground truth, limited only by the amount of data it produces.

Currently, the algorithm only works as expected if the voxel grid has a larger extent than the mesh for which the ground truth should be computed because we assume that each ray starts in the state *outside*. To work with arbitrary voxel grid and mesh combinations we need to introduce a check for the starting point of each ray whether it is *inside* or *outside* the given mesh.

The realistically simulated data set. In the first run we use our fully automated simulation pipeline to simulate 675 realistic CT scans for training. This corresponds to 25 virtual cast parts, each of which is simulated with 27 different artifact gradations of the artifact space. With these CT scans we simulate about 420 000 *defects*. Each CT scan has a size of 1000 × 1000 × 1000 voxel and is stored as 16 bit unsigned integer. Therefore, the complete training set has a size of 1.35 TB. The validation set is generated in the same manner from two additional virtual die casts to yield 54 CT scans. These CT scans have the same size as the CT scans in the training set, which adds another 0.11 TB to the data set. Simulating training and validation data takes about two weeks, using four Intel Xeon E5-2687 CPUs with 10 cores each and hyper-threading. That is quite a long time, considering that other approaches like domain randomization or generative models yield their data in next to no time. However, the robust results and the trust of domain experts justify the increased effort for data preparation. Figure 3.19 provides a brief overview of how the simulated data looks like compared to patches taken from real CT scans.

A huge benefit of this simulation pipeline is its flexibility to be easily adapted to other applications. All we need is a mesh and some material specifications. For example, we could extend the pipeline to simulate data for the inspection of fiber composite materials. Here, hand-labeling can be even harder because the tiny fibers and the surrounding matrix often consist of materials with similar absorption coefficients. With the simulation pipeline we just need to place meshes of fibers in a virtual matrix material and can compute a precise ground truth. Another possibility is the detection of specific parts in a CT scan of assembled objects. With the simulation pipeline we simulate, for



(a) A model of a set of teeth based on [159]. They form the basis of the simulation.

(b) A simulated dental cone-beam CT scan with the mesh (orange outline) as precise ground truth.

(c) A real dental cone-beam CT scan [160] with the results of the prediction model as orange overlay.

Figure 3.20.: We can adapt the simulation pipeline to create realistic simulations of other fields of application of computed tomography. For example, with we can simulate dental cone-beam CT to produce data for precise segmentation of teeth in low-dose CT scans.

instance, a given set of screws in a broad variety of different background materials. The trained segmentation model then should be able to find these screws in the assembled part. Furthermore, it can be extended to serve in the field of medical CT. The challenging aspect of medical data is that the variation in shape matters. For example, when segmenting organs in a CT scan of a human body, they all have a distinct shape which, however, varies from human to human and the organs are packed tight but may not overlap. This needs to be considered when creating the meshes.

We demonstrate the potential for medical analyses by means of *segmenting teeth* in dental cone-beam CT. To simulate our dental data we utilize existing meshes of teeth [159] and add the root canals and the hollow jawbones (see Figure 3.20a). Then, we simulate a scan using 110 kV at 0.5 mA using a coarse detector of 512×512 pixel with a pixel size of 400 µm×400 µm. The source and detector are 500 mm apart, the virtual head is placed centered between source and detector (at a distance of 250 mm to the source). We only make 180 projections and use an exposure time of only 0.1 s per projection. An example of the simulated training data is shown in Figure 3.20b. Finally, we evaluate the trained segmentation model on real dental cone-beam CT data [160]. As this data comes without a hand-labeled ground truth, Figure 3.20c shows a qualitative example of the results. These look quite promising, however, the model sometimes confuses parts of the jawbone with teeth and struggles if amalgam crowns are present in the scans as they are not part of the training set. Nevertheless, these simulations offer the opportunity to fine-tune pre-trained models on realistic simulations of the jaw of a specific patient. Thus, the models can be fine-tuned to cope with severe image artifacts which arise from *low-dose CT*. This further allows to reduce the dose for the patient during the actual CT scan.

4. Reference-free Defect Detection

Our goal is to create a novel defect detection and segmentation algorithm which not only outperforms state-of-the-art methods but is also invariant to changes and adaptations in the actual shape of the part under examination. Reference-based approaches can, therefore, not be considered because they need a "golden part" that exactly matches the examined parts. Due to its successes in other semantic segmentation tasks [14, 15, 22–25], deep learning is promising to be as well applicable in the field of non-destructive testing, more precisely the defect detection (and segmentation) in CT scans. Whereby, we initially focus on CT scans of cast aluminum parts. So far, in literature deep learning is mostly used to artificially improve the data quality of the CT scans to ease the inspection with traditional methods. This is done by reducing the strength of image artifacts like scatter [52], metal artifacts [161], or noise due to a low dose CT [50,51]. In contrast, we develop a deep defect detection model which is able to cope with the image artifacts in the CT scan and directly focuses on the task of semantic segmentation. Hence, we need to clarify two major issues: On the one hand, we need to check whether our realistically simulated data is good enough to serve as training data for machine learning methods—and deep learning methods in particular. On the other hand, we need to check if deep learning really is the best approach for this task or if traditional machine learning methods or even plain image processing techniques are more suited.

Our requirements for a reference-free defect detection algorithm, i. e. an algorithm that needs no prior knowledge about the exact part under examination (compare to Section 2.2.2), are the following: (1) Firstly, the results need to be precise and consistent to enable reliable decisions. (2) Secondly, the algorithm needs to be able to deal with the increased artifact-levels which we encounter, for example, in in-line scenarios. (3) Finally, the inference time needs to be short enough so that the algorithm can keep up with the high throughput in in-line scenarios. In this chapter we explore different methods for each of the three categories (i) image processing techniques, (ii) traditional machine learning, and (iii) deep learning, and check their suitability. We discuss the benefits and drawbacks of each method and carefully select the three most competitive ones to form the sound baseline which we use for further comparison: (i) a filter-based anomaly detection approach using morphological filters, (ii) a sliding window approach comprising densely computed filter-based features and a random forest, and (iii) an end-to-end trainable two-step fully convolutional network using an inspection step with a high recall and a downstream refinement step with a high precision output.

At this point we like to note that even though there are commercial methods available which would mostly belong to the category of image processing techniques and achieve really good results, we do not include them in our study to maintain full transparency and not rely on black box models which merely allow us to tune some parameters but do not reveal any insight in their actual implementation.

4.1. Image Processing Techniques

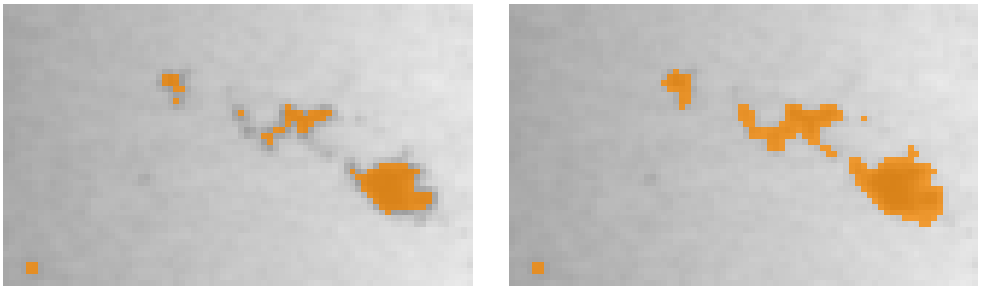
The idea of image processing is to carve out the target class so that it can be separated from the background by applying a simple threshold on the processed image. As opposed to learning-based techniques, the choice of individual processing steps and the choice of how to concatenate these steps into an image processing pipeline is made by the human designer of the algorithm. Often, the algorithms are based on precisely formulated mathematical concepts such as averaging and derivatives or draw inspiration from physics, e. g. by applying differential equations to gray-values. An important consideration is that the pipeline of image processing steps can be explained to "make sense" for the application domain to other experts. These methods often compute statistics of the image, operating within a small locally restricted neighborhood, i. e. the window, which is then sled across the entire image. Examples are the smoothing with Gaussian filter kernels, i. e. computing the mean weighted by the distance [87], or the edge detection with Sobel filter kernels, i. e. computing the discrete gray-value gradient along a given axis [87]. Beyond that, there are more sophisticated non-local techniques. For example, for the denoising of images, there is the non-local means [162] (which actually is implemented semi-locally [163]) or the computation of the total variation [164, 165]. Both methods reduce the overall image noise while preserving as much edge information as possible. Depending on the chosen parameters, these denoising methods, however, also remove smaller defects from the CT scan. Therefore, we shy away from pre-processing the images and rather look for methods which are capable of dealing with image noise. For reference-free defect detection we take a closer look at (i) locally adaptive thresholding [166–168], which chooses the threshold in relation to the contrast within a local neighborhood, (ii) template matching [49, 84, 85, 169], which convolves the image with an quintessential pattern of the target class, and (iii) a morphological method [49, 170, 171] which aims for an anomaly detection approach by creating a "defect-free" part to evaluate the deviations.

4.1.1. Adaptive Thresholding

Simply applying a *global threshold* to the plain gray-value image is doomed to fail [87, 168]. Due to cupping artifacts which arise from the effects of beam hardening and scatter, we are usually never able to find a globally optimal threshold t_{opt} which separates all defects from the material in all regions. While a chosen t_{opt} might be good to segment defects which are close to the outer material boundaries, the segmentation can become more fragile in regions closer to the center of the CT scan because there typically is less contrast due to cupping artifacts. *Locally adaptive thresholding* techniques try to cope with local changes in image contrast by taking local statistics within a given region, i. e. the window or *neighborhood*, around each voxel into account [166–168].

Following Equation (4.1), we implement our locally adaptive threshold by means of comparing each voxel to the weighted sum of its neighborhood. The binary segmentation mask B at a position p which is denoted by its coordinates x, y, and z is either 1 if the input image I at p is smaller than the locally defined threshold $t_{x,y,z}$ or 0 otherwise.

$$B(x,y,z) = \begin{cases} 1, & \text{if} \quad I(x,y,z) < t_{x,y,z} \\ 0, & \text{otherwise} \end{cases}, \text{ with } t_{x,y,z} = \sum_{i,j,k} \left(\omega_{i,j,k} I(x+i, y+j, z+k) \right) - c \tag{4.1}$$



(a) Using a global threshold we have to weigh up receiving false negatives in the low contrast regions of the image against precisely segmenting the defects in the high contrast regions.

(b) Using a locally adaptive threshold we obtain a precise segmentation in the low contrast regions as well as in the high contrast regions. Nevertheless, the method still is susceptible to image noise.

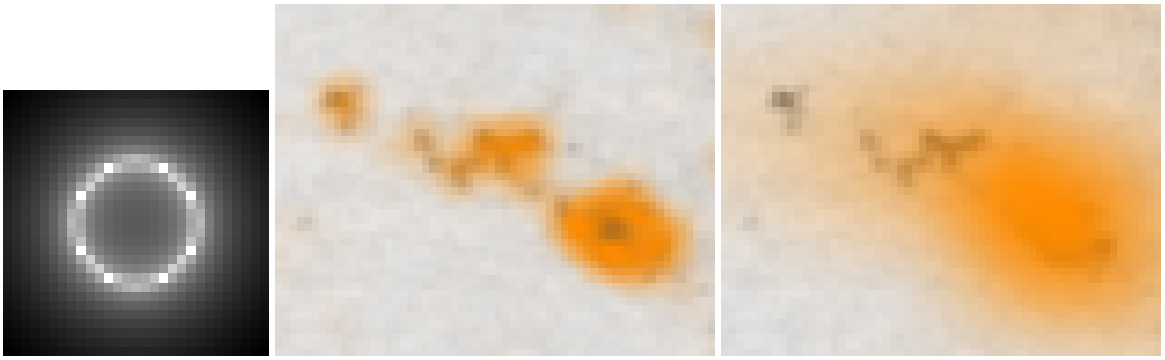
Figure 4.1.: For a global threshold to be applicable, the image needs a constant contrast and a low noise-level. However, the contrast is subject to strong fluctuations due to beam hardening, scatter, and other artifacts. A locally adaptive threshold, therefore, yields more consistent results. The orange overlay shows the binary segmentation mask.

To compute $t_{x,y,z}$ we sum up all gray-values I within the window of given size N centered around p, weighted by $\omega_{i,j,k}$. Whereby, we choose $\omega_{i,j,k}$ to form a Gaussian around p, i. e. we weight the values of I by their distance to the center voxel. For our data set we choose $\sigma = 5.0$ to capture the local contrast. Then, we subtract $c = 0.2$ to obtain the local threshold. In other words: we smooth the image using a Gaussian filter kernel with edge length $N = 6\sigma$, subtract the constant c and then look for values in the image which are smaller than their corresponding averaged value to obtain our segmentation. In Figure 4.1 we compare using a global threshold to using a locally adaptive threshold. The CT scan has a region of low contrast and a region of higher contrast. While the results both look quite good at first sight, we see that by choosing a globally optimal threshold (Figure 4.1a), we have to use a lower value to avoid false negative responses in the region of low contrast. This, however, has an impact on the region of higher contrast: here, we lose portions of the defects in their boundary region—the segmentation becomes less precise. In contrast, with a locally adaptive threshold (Figure 4.1b), we are able to avoid false positive responses in the region of low contrast and completely segment the defect in the region of higher contrast.

While this method is very fast and would allow for a high throughput, it, still, has a hard time to yield precise and robust results for everyday inspection tasks. Using a locally adaptive threshold mitigates the effects of local changes in image contrast, however, the challenges of image noise remain. Therefore, this method still requires a high scan quality and such demands for long exposure times which are not always feasible. For example, in in-line scenarios every second counts.

4.1.2. Template Matching

To deal with image noise, we need to include the information of several voxels. Therefore, we use a kind of a voting approach, where each voxel within a given neighborhood votes for the center voxel whether it is part of a defect or not. We implement this by means of a *cross correlation* [84] of the CT scan with an abstract defect specimen, i. e. the *template*. The more voxels match the template the more significant the filter response becomes. If the responses of the cross correlation are above a given threshold, we locate a defect instance, i. e. a match.



(a) Spherical template

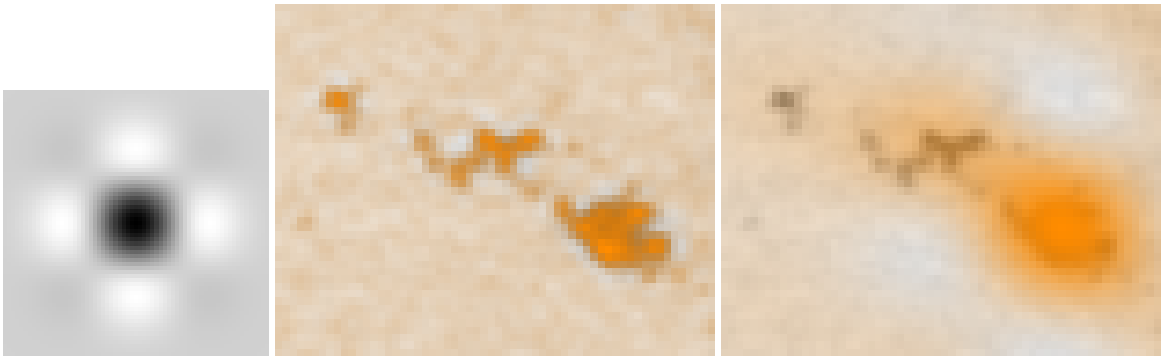
(b) Result of the correlation of the spherical template ($r = 1$) with the gradient magnitude image of the CT scan.

(c) Result of the correlation of the spherical template ($r = 8$) with the gradient magnitude image of the CT scan.

Figure 4.2.: To detect defects in CT scans via template matching we turn towards more abstract shapes, as defects have no definite form. We correlate a spherical template (a) of different sizes with the gradient magnitude image of the CT scan. This method is less prone to image noise, yet, it leads to more fuzzy results (b and c). The orange overlay shows the filter response of the correlation of the gradient magnitude image with the template at different sizes.

This means we need a basic notion of what we are looking for. But, defects do not have a single typical shape—or size. To tackle the shape issue, we turn to more abstract representations. In CT scans, defects usually are characterized by an abrupt change in the gray-values, which exceeds the change induced by image noise, i. e. they have a high image gradient. Therefore, we should be able to find defects by looking for roundish structures within the gradient magnitude image of the CT scan. This further allows for a better treatment of image noise as we do not rely on the gray-values directly. We use the Scharr operator [172] to compute the individual gradients along each dimension and so slightly smooth the CT scan and mitigate the influence of image noise. Moreover, the *spherical template* has the benefit of being rotation invariant so that the orientation of a defect does not lower the filter response. To tackle the size issue, we use a pyramidal approach. We correlate the template with different resolutions of the CT scan. In exchange, the inference time is increased with each additional cross correlation. Figure 4.2a shows the center slice of the three-dimensional spherical template which we use to find defect instances in the gradient magnitude image of the CT scan. In Figure 4.2b we see the result of the cross-correlation of the CT scan of our aluminum part with a spherical template with $r = 1$ voxels. In Figure 4.2c we see the corresponding result when using a spherical template with $r = 8$ voxels. The smaller template also yields high responses in the boundary region of larger defects. For larger template sizes (or smaller resolutions of the CT scan) the results become very fuzzy. This fuzziness prevents a precise segmentation. Therefore, we have to try a different template.

Another abstract template is formed by a combination of different Gabor filters [88]. A Gabor filter is a combination of a sinusoidal wave and a Gaussian. It analyses an image for given frequencies. By merging multiple Gabor filters with different orientations, we obtain a *blob detector* which we can use to find defects directly in the gray-values of a CT scan (see Figure 4.3a). In our case we combine three Gabor filters which are oriented along the three spatial axes of the gray-value grids of the CT scan. The individual filter kernels are combined using convolutions. The filter template has large negative values at its center and is surrounded by large positive values. Due to the combination



(a) Gabor template

(b) Result of the correlation of the Gabor template ($r = 1$) with the gray-value image of the CT scan.

(c) Result of the correlation of the Gabor template ($r = 8$) with the gray-value image of the CT scan.

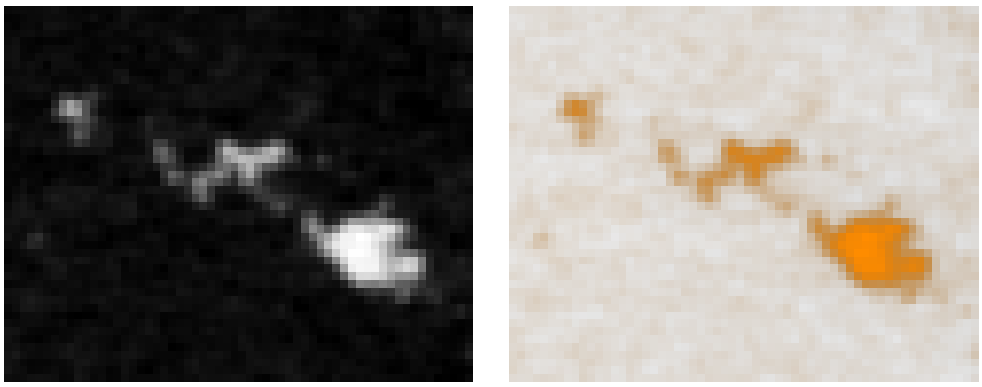
Figure 4.3.: Using a Gabor template (a), the results become less fuzzy but still do not allow for a precise segmentation (b and c). In particular, we observe stronger responses in the noisier flawless parts of the material. The orange overlay shows the filter response of the correlation of the gray-value image with the template at different sizes.

with the Gaussian the values quickly move towards zero with their distance to the center. With defects having lower gray-values than their surroundings we receive low negative values from the center and large positive values from the surroundings, which in total yields a strong response when template and defect align. In Figure 4.3b we see the result of the cross-correlation of the CT scan of our aluminum part with the Gabor template with $r = 1$ voxels. In Figure 4.3c we see the corresponding result when using a Gabor template with $r = 8$ voxels. The results of this approach are less fuzzy than with the plain spherical template, however, they are still too fuzzy to be used as precise semantic segmentation.

Due to their fuzziness, the results of the template matching approaches, however, only allow for an indication of where a defect might be. For a precise segmentation we then have to use another method in the indicated region, which would slow down the computation. This is why template matching filters usually are part of more complex pipelines, for example, to filter the responses of a candidate selection process [49].

4.1.3. Morphological Filters

The strength of reference-based approaches lies in the comparison to a defect-free specimen. By evaluating the differences of the (possibly) defective part and the defect-free reference part we find any deviations. To make this *anomaly detection approach* independent of any defect-free reference part, we need a method to compute the reference from any defect-afflicted part. While a Gaussian filter merely blurs the CT scan until the defects (end every other structure) eventually vanishes, we go for morphological filters [49, 171] which propagate the maximum or minimum gray-value within a local neighborhood. By combining those filters we create a defect-free part from which we subtract the originally defect-afflicted part to detect the anomalies, i. e. to find the defects. Applying a threshold to this residual image yields a relatively precise semantic segmentation of the defects. As this method takes a local neighborhood into account it is less sensitive to image noise than the plain



(a) Anomalies in the CT scan: The larger the gray-value in the difference image, the more we deviate from the generated defect-free specimen.

(b) The difference image allows for a relatively precise segmentation of the defects.

Figure 4.4.: Using morphological filters we create a defect-free specimen, from which we then subtract the original part to obtain the residual image (a) which enables us to find anomalies. Within the material boundaries these usually are defects. The orange overlay shows the difference to the generated flawless part.

gray-value-based threshold. By combining this approach with an adaptive thresholding instead of using a global threshold, we obtain a more robust result which is less sensitive to changes in the image contrast, too.

Common to all image processing techniques is that they require a well defined analysis area. We need to restrict the region in which we apply the analysis to the material of the part—or at least where we expect it to be. Otherwise we obtain false positive results especially in the boundary region of the material and sometimes even in the background.

With the morphological filters, we are able to create a fast defect detection algorithm, which solely builds up on image processing methods. Furthermore, this method is the first step in the candidate selection pipeline which is used in [49] to pre-select possible defects. As it already yields satisfying segmentation results without any post-processing or classification, we include it as standalone method in our further experiments. This method is further referred to as “filter-based method”.

4.2. Traditional Machine Learning

With image processing methods, it typically becomes harder to keep the balance between false positive and false negative responses as the artifact-level rises. Using a second step which filters undesirable false positive responses allows the detection method to be more sensitive but still achieve a high precision. For the filter process, it is beneficial to employ more global statistics (from several CT scans), i. e. machine learning. One way to further subdivide the class of machine learning approaches is by the way the input data is processed. A machine learning algorithm either classifies (i) pre-selected instances, for example, given by a filter-based method [86, 89], (ii) groups of image elements like super pixels generated by clustering algorithms [173, 174], or (iii) individual image elements which in our case are the plain voxels [94, 175]. For (i) the extracted features can be more

complex reflecting the knowledge of domain experts, which tend to be more computationally intensive. For (iii) we need simpler features which can be computed quickly as we have to compute the features for billions of voxels. For (ii) we need to find the right balance of complexity and computational effort. In this section, we evaluate two methods: an instance-based approach which follows a traditional classification pipeline and a voxel-based approach which outputs a probability for each voxel.

4.2.1. Candidate Classification

This method comprises three steps: the candidate selection, the feature extraction, and the actual classification. We follow the "learning-based automatic defect recognition" approach [49], even though the method is tailored towards the detection of roundish gas pores. Hence, we need to make some adjustments for the method to detect flaws of arbitrary shape.

The candidate selection process uses the filter-based method which we describe in Section 4.1.3. The result is then processed with a Gabor filter kernel, i. e. the template matching approach (see Section 4.1.2). This filters most of the smaller false positive responses arising from image noise but also restricts the result to roundish defects of a given size. Applying a threshold to the output of the template matching step gives us the seed points of the candidates. The actual candidate region, then, is computed by comparing the defective part to a precisely aligned defect-free reference part, following a reference-based "golden part" approach. We, however, need a more general and reference-free approach. Therefore, we skip the restricting template matching and use the output of the filter-based method directly as seeds for our candidate regions. Instead of comparing the defective part with a defect-free reference part, we use a locally optimal threshold [176] to refine the candidate region. This should provide us with a candidate selection which is independent of the actual defect shape and the actual part. The subsequent classification should be sufficient to distinguish between false positive responses arising from image artifacts and actual defects.

The feature computation remains unchanged. For each of the refined candidates, we extract the same 29 features [49]. These features comprise 25 gray-value-based features from the candidate region itself and its surrounding as well as 4 curvature-based features of the candidate region. The computation of the gray-value features requires a minimum diameter of 4 voxels and the comparison to a defect-free reference part. For our reference-free adaption, we use the defect-free specimen generated by the morphological filters during the candidate selection. The features are computed as follows:

Gray-value-based features These features are computed by means of the gray-level co-occurrence matrix (GLCM) [177]. The GLCM describes how often a given gray-value occurs as a neighbor of any other gray-value at a given distance d and a given direction θ [177]. As in [49] we compute three GLCMs with $d = 1$, $d = 2$, and $d = 3$ for the candidate region and another three for its surrounding. For each GLCM, we combine the statistics of six directions, i. e. of the six directly adjacent gray-values. (Note that in the original paper 12 directions are used. Unfortunately, it is not stated which ones and there is no common unique way to choose 12 of 26 neighbors.) Figure 4.5 shows three exemplary GLCMs of a defect, an edge, and a flawless region. In addition to the three GLCMs of the candidate region, we compute three GLCMs of the surrounding region of the candidate. To compute the surrounding region we dilate the

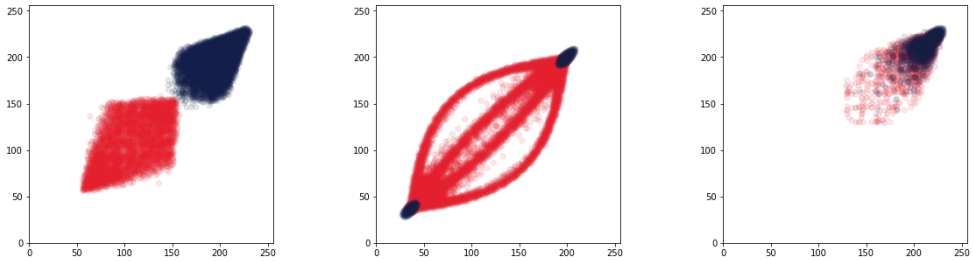


Figure 4.5.: The gray-value-based features are computed from the GLCM. From left to right we show the GLCM of a defect, an edge, and a defect-free sample. In red we show the GLCM of the candidate region and in blue the GLCM of the border region, respectively. The border region is defined by the dilated candidate mask (which is enlarged by 6 voxels) minus the original candidate mask.

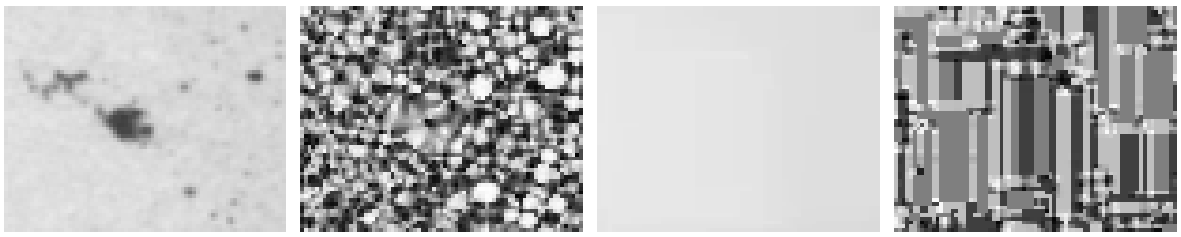
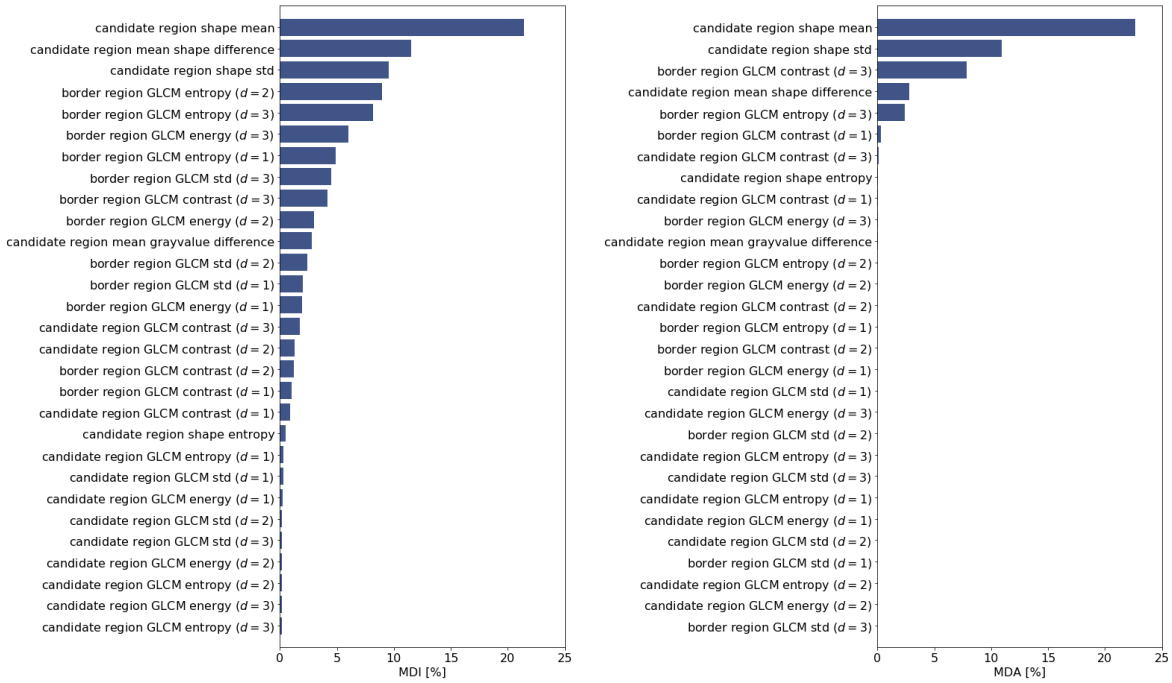


Figure 4.6.: The curvature-based features are computed from the second derivatives of the defect candidate and its flawless counterpart. To be more precise, we compute the scalar shape index from the principal curvatures. From left to right we show the defective sample and the corresponding shape index values as well as the flawless counterpart and its corresponding shape index values.

candidate mask by 6 voxels and subtract the original mask. In the original implementation, they use the same radius for the dilation as they use for the morphological filter. However, if we use larger radii in order to find larger defects, it could happen that there are other flaws in the border region of the candidate we are looking at. Because these defects have the potential to disturb the GLCM of the border region, we fix the radius to 6 voxels. The resulting region should still contain sufficiently many voxels for a meaningful statistic but avoid conflicts with other defects. Per GLCM we compute the energy, contrast, entropy, and standard deviation as features, which are 24 features in total. The last gray-value-based feature is the mean residual value within the candidate region, i. e. the mean response of the candidate selection filter.

Curvature-based features The principal curvatures are approximated by computing the voxel-wise second derivatives. Depending on the principal curvatures [178] a "shape descriptor" is computed for each voxel within the candidate region [49]. The shape descriptor assigns each voxel one of the categories knob, ridge, saddle, cleft, or bag. The mean, standard deviation, and entropy of the shape descriptors form three of the four curvature-based features. However, we find it more useful to use the *shape index* [179] instead of the shape descriptor because it is a more continuous measure without further hand-crafted decision boundaries. The last curvature-based feature is the mean difference in the shape index of the original gray-value image and the defect-free specimen which in our case is generated using morphological filters. Figure 4.6 visualizes the shape index distribution for a defect and the corresponding generated defect-free specimen.



(a) Mean decrease in impurity (Gini importance).

(b) Mean decrease in accuracy (permutation importance).

Figure 4.7.: When measuring the importance of the individual features proposed by [49], we see that most of the gray-value-based features do not contribute much to the output of the individual decision trees of the random forest. Whereby, the GLCM of the border region of the defects contributes more to the class purity within the decision trees than those of the actual candidate region. The most significant features are the mean shape index and its difference to the defect-free specimen. This makes sense as open false positive responses at the material boundary have different shape indices than closed inner defects or noisy particles.

Based on these 29 features a random forest [93], finally, decides whether a candidate is a surface defect, an inner defect, or a false positive response due to an image artifact. The random forest consists of 256 decision trees with a maximum depth of 32 decisions each. For training, we extract the 29 features for each precisely labeled defect of our realistically simulated training set, which meets the required minimal diameter of 4 voxels. Computing 29 complex features takes quite a while. We, therefore, run a feature relevance analysis computing the mean decrease in impurity (MDI) which measures by how much the impurity of the child nodes of a decision tree is decreased when a given feature was involved in the decision [180], i. e. Gini importance (see Figure 4.7a), and the mean decrease in accuracy (MDA) which measures by how much the accuracy is decreased when the values of a given feature are shuffled [180], i. e. permutation importance (see Figure 4.7b). We see that the most important features are the curvature-based features and that most of the gray-value-based features computed from the GLCMs do not contribute much to a proper decision. The mean gray-value difference does not contribute much either. This probably is due to the fact that it is used to select the candidates, which means that all candidates have a minimum mean gray-value difference. If we need to tune the inference speed, we could reduce the amount of features easily.

For testing, we apply the full machine learning pipeline on our simulated validation set and our two real CT scans. The largest defects in our data set have a diameter of 27 voxels. Therefore, we choose a radius of 15 voxels for the filter kernel of the candidate selection. For the local refinement

of the defect boundary, we consider the region of the original segmentation and add a margin of 6 voxels. However, we have quite a hard time to tune the threshold for the filter response which provides us with the initial candidates: If the filter is too sensitive, we get a lot of tiny false positive responses which tremendously increase the computation time as we need to extract the features for each candidate. On the other hand, if the filter is too restraining, we get about the same result for the larger defects as the filter-based method but miss out all the smaller instances. Moreover, we cannot use a fixed threshold for all the data in the validation set. Due to the variance in data quality each CT scan has its individual threshold which suits it best. As the original intent of this method is to serve in in-line (or at-line) scenarios, the threshold usually has to be tuned once and should be applicable to all the following CT scans—which should look similar. To obtain a higher recall, i. e. finding more candidates, we reduce the optimal threshold for the candidate selection by 10 % and see whether we can filter the false positive responses that come along. The optimal threshold is defined in terms of the optimal result of the filter-based method. Adding a filter which removes all instances which do not meet the minimal size after the local refinement, yields more robust results but does not significantly improve the prediction performance.

Our realistically simulated validation set contains 54 CT scans of two parts which have 479 and 459 defects with $d \geq 4$ voxels, respectively. For the CT scans with a high scan-quality, i. e. low artifact-levels, we obtain about 610 and 570 candidates for the two parts, respectively. Out of these candidates about 420 are categorized as defect with a mean probability of 72 %. 410 are correctly identified defects, the remaining 10 false positives mainly occur in image noise. Even though the missing defects are found by the candidate selection, they are discarded with a probability of only 30 %. Here, the classifier is too indecisive. Yet, the misclassified defects cannot be clearly categorized, they range in all sizes and positions in the data set. For the CT scans with a low scan-quality, i. e. high artifact-lavels, we obtain about 420 and 430 candidates for the two parts, respectively. Out of these candidates about 340 are categorized as defect with a mean probability of only 54 %. The number of false positives is reduced to only one or two instances per scan. In total, most of the false positive responses are ruled out, which means we do not need to restrict the analysis area to the material anymore. Nevertheless, we lose a lot of actual positives due to the indecisiveness of the classifier. Sometimes multiple actual defects are merged to one large candidate, sometimes the defects might be too large so that the mean shape index is not as meaningful as it is for other defects, but sometimes this also has no obvious reason. In Figure 4.8 we show a few examples from the simulated validation set. These comprise correctly discarded false positives which occur at screw threads, streaking artifacts or ring artifacts as well as a few examples of misclassified instances.

The classifier itself yields good results, as long as the candidate region fits the actual defect. The extent of the candidate region has a huge impact on the feature computation. A drawback of an instance-based method is that the defect is either in the final segmentation mask as it is or it is removed completely. This becomes more severe if the classifier relies on a precise segmentation—which cannot be guaranteed in an artifact-afflicted CT scan. If, for example, the candidate region is too wide, it contains too many defect-free voxels and changes the appearance of the GLCM and shifts the mean curvature more towards defect free samples. We, therefore, test the classifier on the features which we extract form perfect candidate regions. We obtain these regions from the precise ground truth of our validation set. The features are extracted from CT scans of defective and from CT scans of defect-free parts which are scanned with the same parameters. For each set of features of a defect there is a set of features from its defect-free counterpart. Here, the classifier achieves an

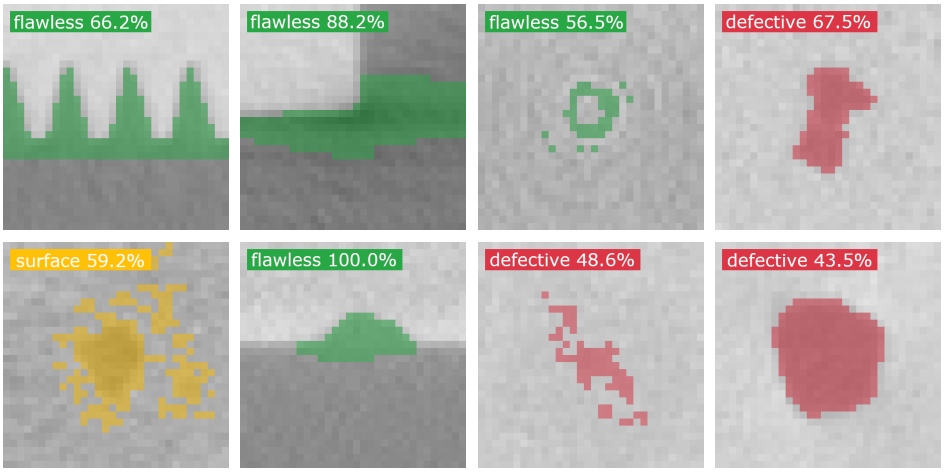


Figure 4.8.: Results of the candidate classification: In the top row we see that most of the common false positive responses of the candidate selection which, for example, occur near screw threads, in streaking artifacts, or in ring artifacts, are correctly classified as flawless. In addition, we obtain high defect probabilities even for those instances which are not roundish. However, there are many false classifications, too (bottom row). Inner defects are classified as surface defects, surface defects are mistaken to be flawless, and especially in noisy regions noise particles are classified as defect. Additionally, some defects have a quite low probability of being a defect despite the roundish appearance. Notably, this applies to larger defect instances.

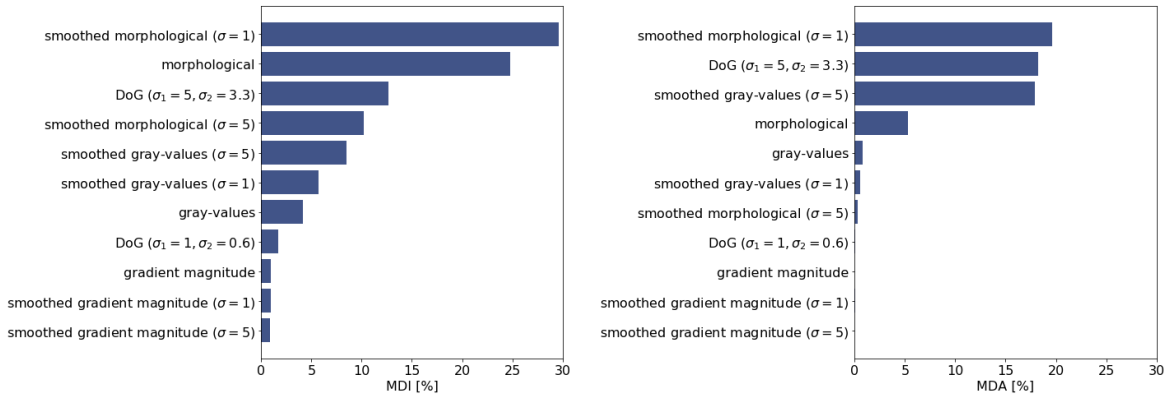
accuracy of 99.25 %.

Due to the complexity in the tuning of the parameters, the restrictions of the minimal diameter, and the dependence on a precise candidate selection, we abandoned this approach. It probably is quite useful for the exact application it was designed for, but we find it very hard to generalize to our needs.

4.2.2. Sliding Window Approach

For this approach, we move from an instance-level classification to a *voxel-level* classification. We extract the features for each voxel in the entire CT scan and predict for each voxel the probability of being part of a defect by sliding a random forest across the CT scan. This method is inspired by the pixel classification work flow of ilastik (European Molecular Biology Laboratory) [94], which is, for example, used to segment cells in neuro-biological images. Furthermore, a similar approach is used to segment objects of daily life [175].

Typical hand-crafted features which are computationally efficient are, for instance, the plain gray-values of the CT scan, the gradient information like direction or magnitude which is extracted using Sobel filters [87], and the DoG filters [83] which act as blob detectors. Moreover, we evaluate the output of the filter-based method as feature. On top of that, it would be conceivable to use the information from the second derivatives like the structure tensor [87] and other more complex features like SIFT [90], SURF [91] or HOG [89]. However, they are more time-consuming to compute and might exceed the strict time constraints we encounter, for example, in in-line scenarios. To reduce the features to a minimum of the most important ones, we again evaluate the MDI and the MDA [180] (see Figure 4.9a and Figure 4.9b, respectively). The most important feature is the output



(a) Mean decrease in impurity (Gini importance).

(b) Mean decrease in accuracy (permutation importance).

Figure 4.9.: The most important features are the responses of the filter-based method and the DoG filter, as they provide a clear indication of where to find defects. The gray-value features seem to only provide guidance to distinguish defects from false positive responses. While we include the gradient magnitude with the intention to precisely define the boundary of a defect, theses features seem to be rather unimportant to the random forest.

of the filter-based method, which is equivalent to the difference to a generated flawless part, and the filter response of the DoG with a large kernel size, i. e. the blob detector. This seems legit as the filter-based method serves as a defect detection method on its own. The other features then improve the distinction to false positives. While the gradient magnitude information seems important to determine the boundaries of a defect, it only contributes little to the decrease in impurity of the nodes. The plain gray-values probably are less important as their information is included in the difference image which we obtain from the filter-based method.

For our production-level classifier, we select the following eight features:

Gray-value First, we compute the weighted mean of the gray-values using two Gaussian filters to capture local effects of image noise (see Figures 4.10b and 4.10c). We compute the Gaussians with $\sigma = 1$ voxels and with $\sigma = 5$ voxels, respectively. We do not include the plain gray-values in our set of features as they are encoded the difference image of the filter-based method.

Gradient magnitude We use the mean gradient magnitude to include information about the defect boundaries. Again, we compute the mean via two Gaussian filters with $\sigma = 1$ voxels and $\sigma = 5$ voxels, respectively (see Figures 4.10d and 4.10e).

Difference of Gaussians Then, we include two difference of Gaussians features [83]. The first DoG is computed using $\sigma_1 = 1$ voxels and $\sigma_2 = 0.6$ voxels and is designed to detect small defects (see Figure 4.10f). The second DoG is computed using $\sigma_1 = 5$ voxels and $\sigma_2 = 3.3$ voxels, respectively, and designed to detect the larger defects (see Figure 4.10g).

Morphology Finally, we use the output of the filter-based method which computes the difference to a defect-free specimen which is generated by applying a morphological filter. Additionally, we compute its local mean using a Gaussian filter with $\sigma = 1$ voxels (see Figures 4.10h and 4.10i).

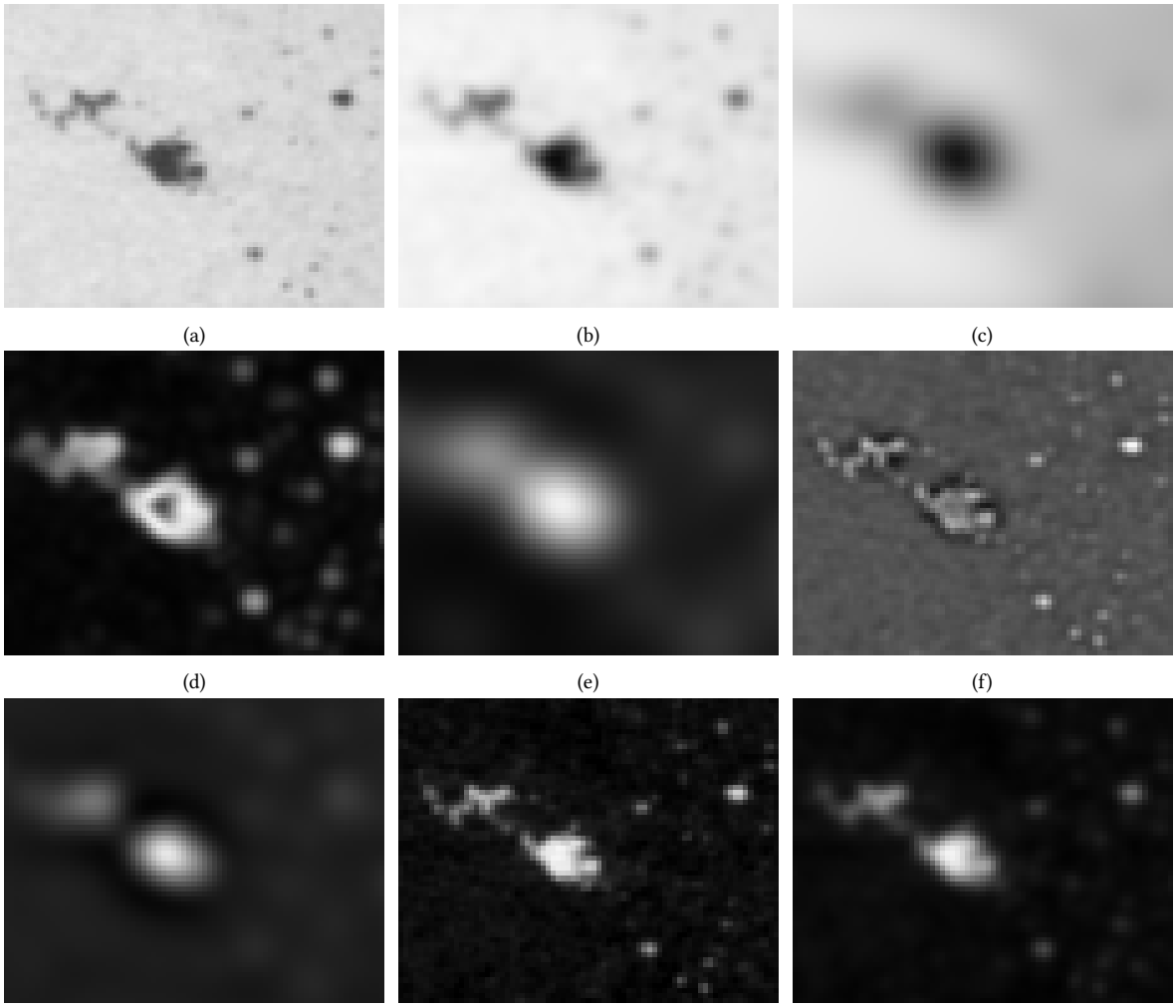
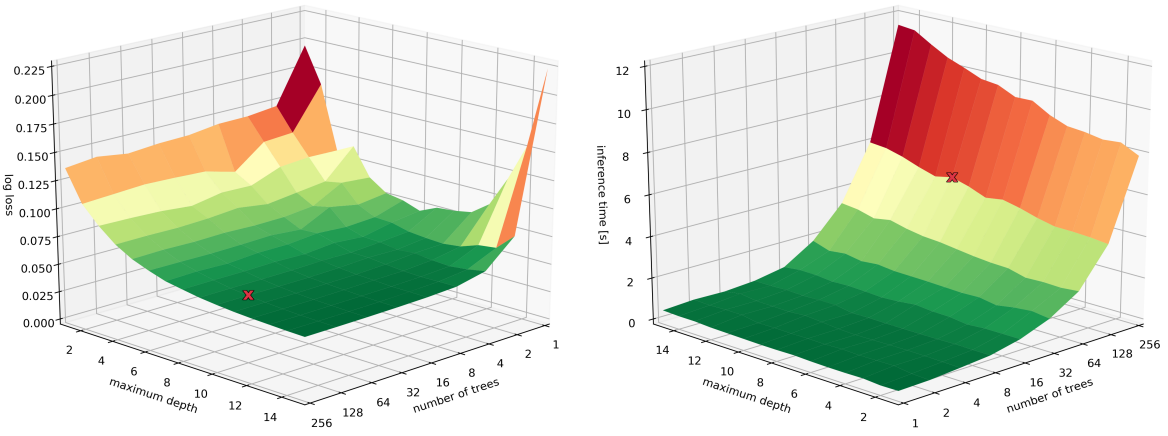


Figure 4.10.: In (a) we show the original gray-values of an exemplary defect and the according filter-responses follow along: (b) The mean gray-values, computed by means of a Gaussian with $\sigma = 1.0$ and (c) a Gaussian with $\sigma = 5.0$. (d) The mean gradient magnitude, computed by means of a Gaussian with $\sigma = 1.0$ and (e) a Gaussian with $\sigma = 5.0$. (f) The DoG with $\sigma_1 = 1.0$ and $\sigma_2 = 0.6$ and (g) with $\sigma_1 = 5.0$ and $\sigma_2 = 3.3$. (h) The difference to a defect-free part computed by means of the filter-based method and (i) the mean difference image computed by means of a Gaussian with $\sigma = 1.0$.

Using all voxels of our realistically simulated data set for training would be an exaggeration. However, when reducing the amount of training samples, we need to carefully sample the training instances; a mere random sampling would not cover all the corner cases. Therefore, we come up with a *sampling strategy* to select the most valuable data points for training. We randomly choose training samples (i) from the defects and (ii) their surrounding region, within a 6 voxels dilation radius. This covers the interesting parts containing the transition from defect to material. Secondly, we (iii) randomly sample around the material boundary to distinguish actual defects from the material boundary and screw threads in particular. The region from which we take our training samples is computed as the difference of the dilated surface mask and the eroded surface mask. The radius



(a) Log loss

(b) Inference time

Figure 4.11.: To decide on how to reduce the inference time of the random forest, we evaluate the log loss, i. e. how much do the predicted probabilities differ from the ground truth, while varying the number of trees in the random forest and their maximal depth. Furthermore, we have a look at the inference time itself. We see that the log loss reaches a plateau as we reach a depth of about 10 decisions per tree. For a low number of trees we observe a peak of high loss values for deeper trees, which indicates an overfitting to the training set. The depth of the trees seems to be less significant for the inference time. However, reducing the number of trees by half also reduces the inference time by half. Note that for reasons of legibility we switch the point of view in the two plots. The red cross marks the configuration we use for our random forest: 128 trees with a maximum depth of 10 decisions.

for the dilation and the erosion are both 6 voxels. Then, we (iv) sample around the rotation axis to explicitly tackle the influence of ring artifacts. Finally, we sample (v) randomly in the material and (vi) in the background. In total, we extract 27 million samples for training of which 3 million are taken from defects.

It is conceivable to further reduce the amount of features without a significant drop in accuracy, e. g. by removing the least important features (here: the gradient magnitude features). However, compared to the classification part, the feature extraction is way less time-consuming. The only concern that would suggest to reduce the number of features is the huge memory consumption: we need eight times the memory to compute and to store the features. The block-wise processing of the input volume allows us to reduce the memory footprint of the computation. We, therefore, rather use more features and tune the properties of the random forest to gain inference speed. The inference speed of a random forest is mainly defined by the number of decisions that have to be made. Thus, we need to limit its capability in favor of inference speed. To restrict the number of decisions we can reduce the number of decision trees in the random forest (which further limits the dynamic range of the output) or the maximum depth of each tree (which limits the decisions within each tree and with that their precision). The optimized classifier not only should be precise in its final decisions but also yield confident predictions with a high probability. In consequence, we evaluate the log loss and, of course, the inference time while varying the number of decision trees and their maximum depth performing an extensive *grid search* (see Figure 4.11a and Figure 4.11b, respectively.). When increasing the depth of the decision trees the log loss gets smaller, i. e. the predictions become more confident. For a low number of decision trees, however, we observe an increasing log loss for deeper trees. This indicates an overfitting to the training data. The inference time is halved as we halve the

number of trees in the random forest. Nevertheless, reducing the maximum depth of each decision tree is less significant. We choose to use 128 decision trees with a maximum depth of 10 decisions. This provides us with a sufficient dynamic range of the probability output and the random forest can develop a high confidence in its results without overfitting.

The voxel-classification is far less artifact-prone than a mere image processing method and, moreover, does not rely on a precise and robust candidate selection process. After tuning the performance it takes about half an hour (1800 s) on a machine with two Intel Xeon E5-2687 CPUs to process a CT scan of 1000 × 1000 × 1000 voxel, of which the computation of the features only takes about 300 s. This is marginally acceptable also for in-line scenarios. We, therefore, include this approach in our further experiments. This method is further referred to as "traditional machine learning method", or simply "traditional method".

4.3. Deep Learning Defect Detection

One of the benefits of deep learning approaches is that neural networks learn to extract the necessary image features on their own. Therefore, there is no need to design and tune hand-crafted features. In exchange, this increases the number of free parameters and we need more data which covers all essential aspects of the task. This shifts the effort from designing features and algorithms more towards the design of the training data. Furthermore, we need to tune the hyper-parameters of the architecture of the model and the training process to ensure that the trained model generalizes well to unseen (real) data. The architecture of a model describes how to combine the different types of layers to a meaningful entity, how many feature channels to use per layer, how big the filter kernels are, and so on. For the training process we can choose from different loss functions which provide direction for the training. We can apply different regularization functions to ensure that the model learns meaningful features and generalizes better, and we can select different optimizers and their parameters. Luckily, there already exists a broad exploratory basis and some best practices we can start off. In this section we describe and compare different model architectures, explore different ways to obtain a deep learning defect detector, and present our curriculum learning approach which we use to speed up the convergence of our models.

4.3.1. Choosing an Architecture

Over the past years a vast amount of different layer types and different architectural components have emerged and have been stacked together to form more and more complex model architectures (see Section 2.3.1). For example, for classification tasks, there are AlexNet [58], VGG-Net [70], ResNet [69], and plenty of other variations of these models. Most authors agree that deeper models are better [181]. The depth, however, is limited by the amount of memory available on the GPU; the problem becomes worse when turning to three-dimensional data. In addition, the deeper the model, the longer are the inference and training times. As we like to obtain a per-voxel output, we need a fully convolutional network (FCN) [12, 18] for the best performance. As mentioned before, object detection architectures like YOLO [75], SSD [182], or Faster R-CNN [74] are not suited for this task as we require a precise segmentation mask instead of a bounding box. Furthermore, the number, shape, and size of defects as well as the size of the input CT scans are widely varying, which is

why we neither can make use of the bounding boxes predicted by combined architectures like Mask Faster R-CNN [76] or the Retina-U-Net [183]. Hence, we start with a promising, basic FCN, namely the U-Net architecture [18], which considers the information of the down-sampling layers during the up-sampling process to achieve a more precise per-voxel output, i. e. the skip-connections. This kind of model already is successfully applied to medical CT data [18, 21, 183–185]. Step by step, we add more complexity and make minor adjustments where we see the necessity to do so, to evolve our two-step defect detection architecture. In this section, we consider increasing the capability of the model, optimizing the convergence during training, providing the necessary context information, and improving the generalization of the model.

The first adjustment we make to the model is replacing the maximum pooling layers by *strided convolutions* [68]. In the original U-Net architecture, pooling layers are used to reduce the spatial resolution. This widens the receptive field of the model in order to gather more context information along the dimensions of convolution. Instead of always choosing the maximum output, we can train a filter kernel on how to combine the outputs of the previous layer. But instead of shifting the filter kernel voxel-wise over the input, we skip every second voxel to reduce the spatial dimensions. An advantage of these strided convolutions over maximum pooling layers is that we enable the model to choose the most appropriate output for its task and that the implementation is computationally more efficient.

An essential detail of the U-Net architecture are the *skip-connections* which are intended to transport the detail information about small structures from the down-sampling part of the model to the up-sampling part [18]. The idea of skip-connections can also be found in the original paper proposing fully convolutional architectures [12], but there are two major differences between both approaches: In [12] the skip-connections sum up the predictions which are made at different scales and use them as input for the next up-scaling layer, while in [18] the feature maps of the down-scaling layers are directly concatenated to the output of the corresponding up-scaling layer. Adding the output of one convolution to the output of another convolution can be seen as a "correction" of the output. This is, for example, done in residual layers [69]. Concatenating the output of two convolutions, however, is more flexible, as the subsequent layer can learn how to combine and fully utilize the input. In the DenseNet architecture [186] this principle is taken to the extreme. Using a fixed function to combine both outputs results in less trainable parameter, i. e. a smaller model, and is faster to compute. We evaluate both approaches: While there are studies suggesting that both approaches yield equivalent results [187], we find that for our task the results of the concatenating model are slightly better. Hence, we stay with the concatenation of the skip-connections.

The skip-connections further contribute to an improved transfer of gradient information back to the down-sampling layer. An improved gradient flow contributes to a faster convergence. We further support the gradient flow by using leaky ReLU [23, 66] activation functions instead of the standard ReLU [65]. These activation functions allow a reduced amount of information to pass if the output of the preceding layer is negative. A standard ReLU would clamp these values hard to zero. Therefore, leaky ReLUs not only contribute to a faster convergence but further tackle the issue of "dying" neurons. A large gradient can cause a neuron to never output any positive values again. Thus, its output—and the gradients passing the neuron—will always be clamped to zero. To further increase the capability of the model, we replace each convolutional layer of the original model by a residual layer [69] which comprises two convolutions and adds its output as "correction" to its input. Our final change to provide more gradient information to leverage the training process is adding *deep*

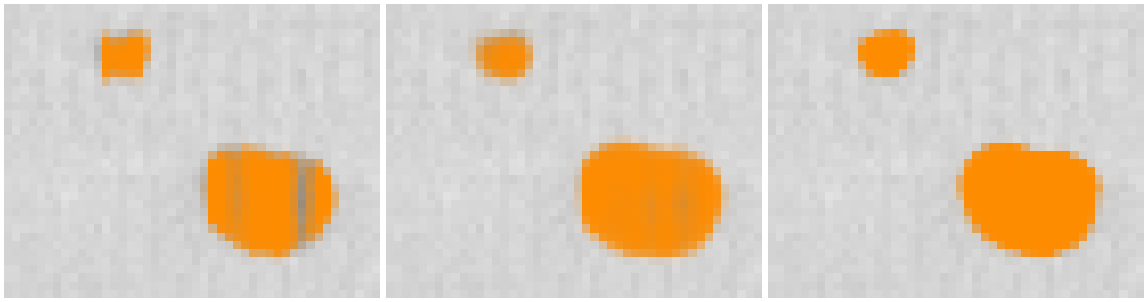
supervision [20, 188]. For each intermediate up-sampling step, we add another transposed convolution to make a prediction based on the available information. The fractional stride of the transposed convolution is chosen to yield a prediction which has the same size as the final output. These predictions are taken into account when computing the loss function. This allows us to directly inject more gradient information into the deeper layers of the network.

Another crucial aspect when designing a model architecture is the amount of *context* information we provide [189]. This not only includes the height and width of the receptive field, which is defined by the kernel sizes and pooling layers but also the data we provide along the third dimension. The original implementation of the U-Net operates on single two-dimensional slices only. Hence, there is no context information along the third dimension. While this model is reported to achieve good results on medical CT data [18, 21, 183–185], it struggles to yield consistent results on our industrial CT scans (see Figure 4.12a). Maybe this is the case because medical data usually has primary direction which often has a different resolution [34]. Maybe this is the case because the anatomy of the human body, the shape and relative position of the inner organs, is rather fixed and does not change much between humans, whereas in industrial data we have to deal with human made structures which are way more arbitrary. Especially when we train a more general model which is not tailored towards a specific scenario. Combining multiple slices to a multi-channel input, i. e. a thick slab, is a simple way to include more context along the analysis direction. In [190], for example, the thick slabs are converted to a two-dimensional slice using a three-dimensional convolution, while we use the thick slab directly as input for our model. With the additional context of only two adjacent slices, i. e. a slab thickness of three, the inconsistencies along the analysis direction already begin to vanish (see Figure 4.12b).

Nevertheless, only when switching from two-dimensional convolutions to full three-dimensional convolutions we gather the full context information within the three-dimensional receptive field. We are able to capture the entire defect and have no more inconsistencies in the analysis result (see Figure 4.12c). The benefit of using single slices as input is, of course, the increased inference speed: On a $1000 \times 1000 \times 1000$ voxel CT scan the single slice U-Net needs about 180 s–200 s to compute a result, while with the full context and three-dimensional convolutions it takes about 500 s–540 s (measured on a NVIDIA TITAN RTX).

A different approach avoiding expensive three-dimensional convolutions would be to run individual predictions on, for example, the three axis-aligned slices and then combine their results. The combination may either be by a majority vote, a simple average or even by asking another neural network [21]. This setup, on the other hand, at least triples the inference time as compared to a simple slice-by-slice approach. These run-times would bring us within the realm of our proposed fully three-dimensional convolutions.

Reducing the spatial resolution is one way to gather context information. Dilated convolutions represent another way [13, 191, 192]. We evaluate this in a separate model, the *dilation model*. With the down-sampling introduced by the strided convolutions we loose spatial information which probably cannot be fully restored by the skip-connections. Therefore, the idea of the dilation model is to use a flat architecture without any down-scaling (see Figure 4.13). Gathering the necessary context information via standard convolutions, however, is not possible: We would need too many layers to be able to fit all of them into the limited memory of the GPU—each layer with n channels occupies n times the memory of the input. Dilated convolutions do not sample their input from adjacent voxels



(a) single slice (b) thick slab (c) full context

Figure 4.12.: Predicting the defect probability in a CT scan slice-by-slice (a) introduces severe inconsistencies along the analysis direction (in our case from left to right). By using thick slabs which are composed of multiple slices (in our case three) and, thus, include more context information, these inconsistencies begin to vanish (b). With the full three-dimensional context there are no inconsistencies in the result anymore (c). The orange overlay shows the prediction output of the respective model.

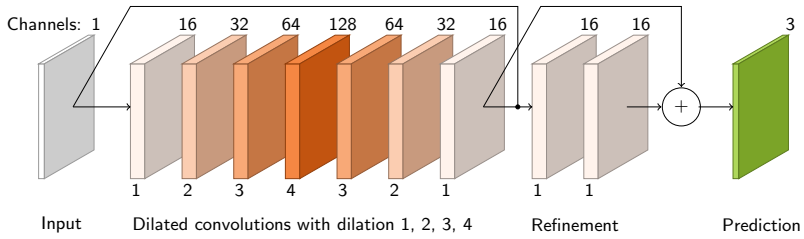


Figure 4.13.: The dilation model: A flat model architecture without any down-sampling or up-sampling. The increasing and decreasing dilation of the filter kernel resembles the context aggregation of the down-sampling layers without loosing spatial information. An additional refinement step should yield a more precise prediction from the extracted features.

but add a spacing between the sample points of their input [191]. With that it is possible to increase the receptive field of a model in fewer layers. This idea arises from the field of remote sensing in which we encounter similar problems as for the defect detection: We need to find very small objects of only few pixels diameter, which can occur in clusters of many instances [192]. Despite the similarity, the smallest instances in the remote sensing images are still larger than the smallest defect instances we need to find. Unfortunately, we find the results of the dilation model to be slightly worse than the results of a model with down-sampling. This is probably due to the limited capability of the network.

Image matting algorithms, i. e. algorithms that separate foreground and background in images, face similar problems as we do: They need to segment fine structures like hair to separate them from the background. To precisely segment structures with only very few pixel in diameter, "deep image matting" proposes a two-stage architecture comprising an encoder-decoder-pair for feature extraction and a refinement stage for an optimal output [193]. Learning from this approach, we add a *refinement step* to our architecture. This step takes all the intermediate results which we have from the deep supervision, the segmentation output of the encoder-decoder-pair, and the original gray-value input into account. The output of the refinement step is then added to the output of the encoder-decoder-pair as a correction. Finally, we add four dropout layers [67] with a dropout rate of 50 % after the four convolutional layers in the final encoding stage. These act as additional regularizer

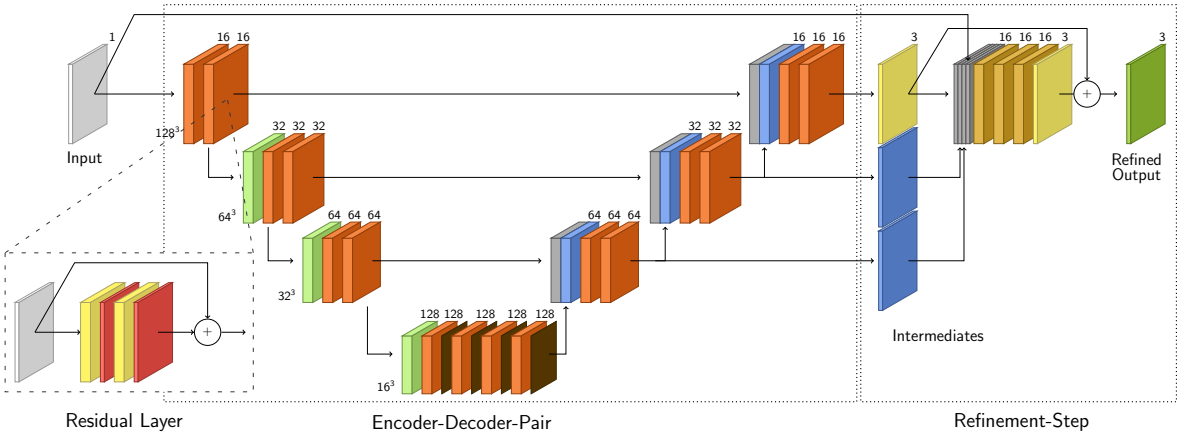


Figure 4.14.: The proposed two-step defect detection architecture combines a U-Net-like encoder-decoder-pair with a refinement-step. The encoder uses residual layers (orange) to create a latent representation of the input (light gray). The residual layers comprise two convolutional layers (yellow) and adds their output to the original input as a correction. After each convolutional layer (yellow) we use a leaky ReLU (red) as activation function. To gather more contextual information and to reduce the spatial resolution we use strided convolutions (light green). These layers also include a leaky ReLU, which is not depicted explicitly. Dropout layers (brown) in the final encoding step further prevent overfitting during training. The decoder up-samples the encoded results using transposed convolutions (blue). These layers also include a leaky ReLU, which is not depicted explicitly. The output of the corresponding encoding layer is integrated in the decoding branch by concatenation (gray). Then, in the refinement-step, all available information is combined using three more convolutional layers with a leaky ReLU (dark yellow) and an additional convolution (yellow) producing the refinement output. Finally, the refinement output is added to the intermediate output yielding the final segmentation mask (green). The number above each layer denotes the number of channels, the number to the left the size of the patches used for training. For all convolutions we use a kernel size of $3 \times 3 \times 3$. The strided convolutions have a stride of $2 \times 2 \times 2$.

to further prevent our model from overfitting. Figure 4.14 shows a schema of the fully modified U-Net-like architecture of our defect detection net. It comprises strided convolutional layers to reduce the spatial resolution and to gather context information, three-dimensional convolutions to fully utilize the available context, and concatenated skip-connections to add detail information during up-sampling. The convolutional layers all have a kernel size of $3 \times 3 \times 3$. Including the bias term they have $27 c_{\text{in}} c_{\text{out}} + c_{\text{out}}$ trainable parameters, where c_{in} denotes the number of channels in the previous layer and c_{out} the number of channels in this layer. The transposed convolutions have a kernel size of $5 \times 5 \times 5$ which increases their number of trainable parameters to $125 c_{\text{in}} c_{\text{out}} + c_{\text{out}}$. In total this model architecture has 6 899 060 trainable parameters. Compared to the about 60 million parameters of AlexNet [58] or the almost 140 million parameters of VGG-Net [70] this number is quite low. However, we only need to detect low-level objects and do not need to distinguish between thousands of categories. Furthermore, increasing the number of filters would require more space on the limited GPU memory. The theoretical *receptive field* of this model covers $191 \times 191 \times 191$ voxel, but the effective receptive field usually covers only a fraction of the theoretical receptive field. Due to the learned weights, not all regions contribute equally to the result which renders the determination of the effective receptive field much harder [194]. The effort to determine the effective receptive field is disproportionate to the benefit, as long it covers the largest defects in our training set, which have diameters of about 27 voxels.

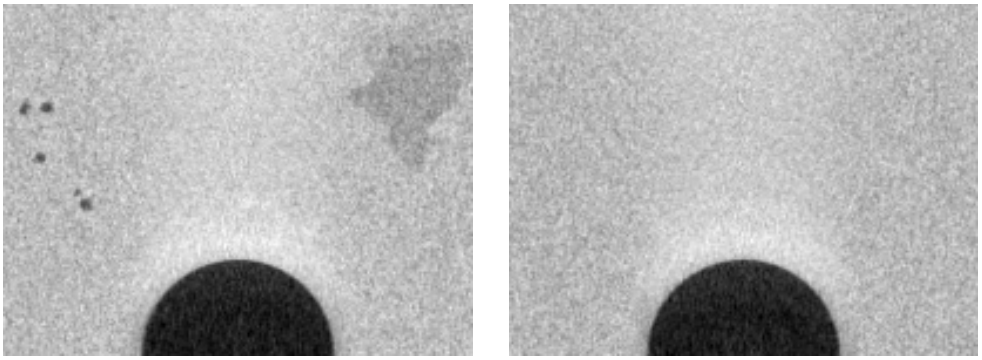
To select the most promising architecture, we compare the three models mentioned above: (i) First, we evaluate a standard U-Net. Here, we process a given CT scan slice by slice along all three spatial dimensions and combine the results using a maximum operation. (ii) Then, we evaluate the flat dilation architecture. This model uses three-dimensional convolutions and, therefore, only needs a single forward pass. (iii) Finally, we evaluate the defect detection architecture, which comprises an encoder-decoder-pair for feature extraction and a separate refinement step. While all three methods yield quite comparable results, method (ii) performs slightly worse than the other methods. This might be due to the limited depth of the model and the inherent limited capability. Method (iii) yields slightly better results than (i) and (ii). Here, the utilization of the full three-dimensional context might be marginally superior to the combination of the three runs of (i). We add a quantitative discussion in Section 5.3.4 as soon as we introduced the necessary metrics. As there is no difference in inference speed between (iii) and (i) we decide to go with (iii), i. e. our defect detection architecture.

4.3.2. Formulating the Target Function

Depending on how we design the output of the model, there exist many different ways to implement a deep learning defect detection algorithm. Each of these approaches requires its own type of labels and its own loss function while the network architecture stays the same. In this section we discuss the following three approaches: (i) The first approach is to resemble the principle of the filter-based method to create a defect-free specimen based on the defect-afflicted part and then to find the defects by evaluating their differences, i. e. an anomaly detection approach. (ii) Alternatively, the model can be used to predict the relative density of the material in the CT scan. The material has a relative density of 1.0, defects a density of 0.0 and occurrences of structural loosening lie somewhere in between. (iii) Finally, the model can be trained to directly yield a segmentation mask, i. e. assigning each voxel a pseudo-probability of being part of a defect or not. This method is a voxel-classification approach, similar to our traditional machine learning method, a typical image segmentation task. The labels, which we create from the precise annotations generated in Section 3.3.3, represent the goal we like to reach and the loss function is designed to guide our model there during training. For each of the approaches we explore what it takes to build a reliable deep defect detector.

Anomaly Detection

With the anomaly detection approach, we basically resemble our filter-based method. However, this time we train a deep neural network to generate the defect-free specimen we need for the comparison instead of simply applying morphological filters. First of all, we need defect-free data as label input for the training process. Thus, we revisit our simulation pipeline and simulate the complete training set once again—this time without any defects (see Figure 4.15b). For the actual training we have two choices: (a) We either train an auto-encoder model by providing only the defect-free CT scans as training input (see Figure 4.15b); or (b) we train the model explicitly to remove any defects from the data by providing the defect-afflicted CT scans as training input (see Figure 4.15a). The idea behind approach (a) is that if the model never sees any defect and only learns how to encode flawless CT scans, it will have no clue how to encode the tiny defect structures and, thus, will generate a defect-free specimen instead [195, 196]. We expect the values of the result to be in the range $[0, 1]$. Hence, we use a sigmoid as activation function of the final layer.



(a) realistically simulated CT scan with defects (b) flawless simulation of the same part

Figure 4.15.: When training our model to generate flawless parts we need flawless parts as label input. Therefore, we simulate our training set again without any defects (b). As input for the model we can choose between the flawless parts (b) training an auto-encoder or the defect-afflicted parts (a) training the model to explicitly erase all defects.

One way to train a generative model to resemble a target image is to use a mean squared error (MSE) loss. For the non-destructive evaluation of fabrics by means of gray-scale images, [195] suggest to use a structural similarity (SSIM) metric as loss function for training and evaluation. According to their analysis, this loss functions directs the model to produce images which optimize the error pattern, i. e. the difference image, by taking local dependencies of neighboring voxels into account. The SSIM takes into account the luminance, the contrast and the structure of two equally sized neighborhoods to compute a distance measure [197]. The structural similarity loss of two patches p and q is defined as shown in Equation (4.2), where μ denotes the mean of a patch, σ the standard deviation, and the constants c_1 and c_2 serve to intercept divisions by zero.

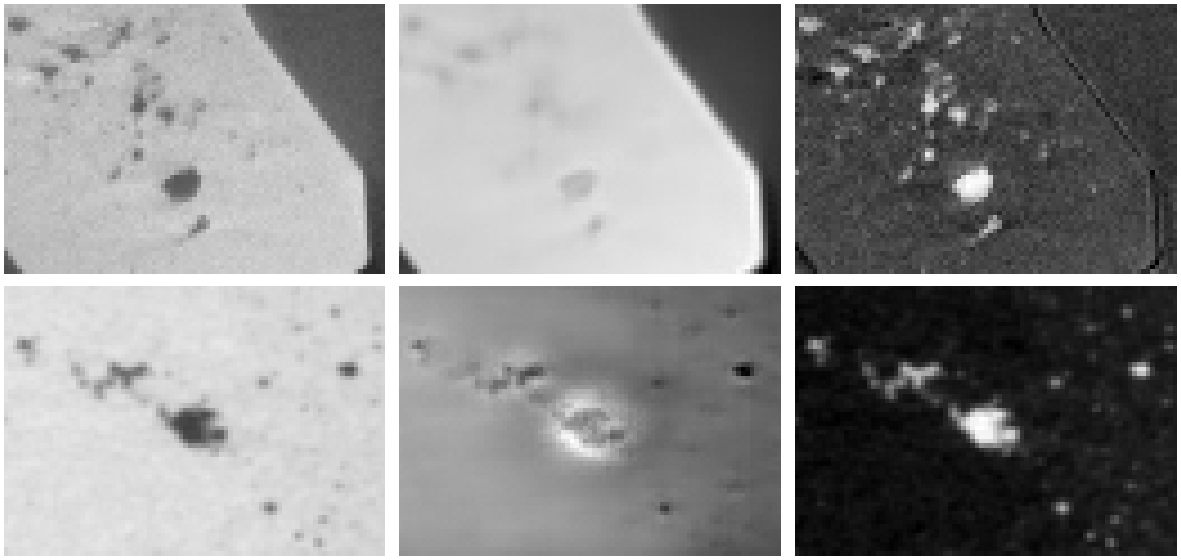
$$\mathcal{L}_{\text{SSIM}} = 1 - \frac{\left(2\mu_p\mu_q + c_1\right)\left(2\sigma_{pq} + c_2\right)}{\left(\mu_p^2 + \mu_q^2 + c_1\right)\left(\sigma_p^2 + \sigma_q^2 + c_2\right)} \tag{4.2}$$

Figure 4.16b shows the model output of the model trained according to method (b), using the SSIM loss. The generated defect-free specimens do not look very convincing. Indeed, the model is not optimized to produce realistic flawless images but to generate images which improve the structural similarity to a flawless part. Therefore, more important are the difference image which we obtain by subtracting the original image from the generated image (see Figure 4.16c).

Despite the difference images looking quite promising, the results do not meet our expectations. Image noise remains an impeding challenge preventing a precise semantic segmentation. Additionally, this method requires to restrict the analysis area to the material part in the CT scan. Therefore, this method seems to be more suited to produce indications of defects, instead of a precise segmentation.

Regression Learning

The anomalies which we try to detect and segment have one thing in common: they all have a density which is different from the actual material of the part. Consequently, we train a regression model

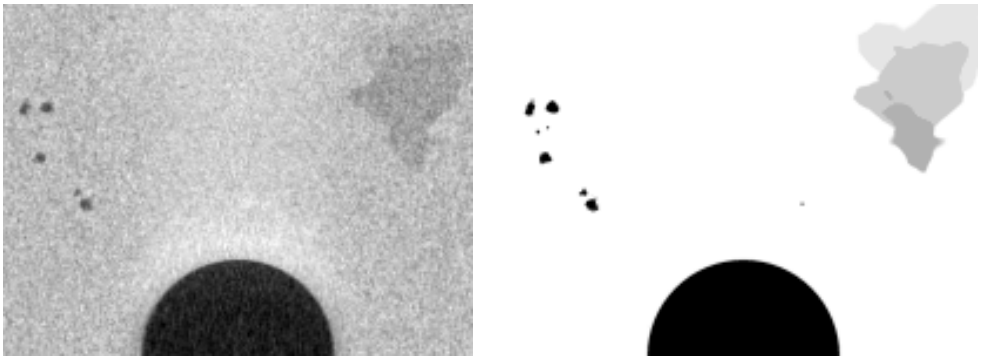


(a) original gray-value images (b) generated defect-free specimens (c) differences to the original CT scan

Figure 4.16.: Anomaly detection: The generated defect-free specimens (b) do not look realistic. However, the model is trained to generate images which maintain the structural similarity to a flawless part within a local neighborhood. The difference images (c), therefore, look more promising but still contain a lot of noise which impede a precise segmentation.

to predict the material density in each voxel. However, CT does not measure the density directly but a relative attenuation coefficient. The measured value further depends on the polychromatic spectrum and the total penetration length. Due to physical effects like beam hardening two voxels which represent material of the same density can have completely different gray-values. Furthermore, artifacts have a huge influence so that the gray-values vary, even though the material has a homogeneous density. Thus, training the model to predict the actual density of $2.80\,\mathrm{g\,cm^{-3}}$ of the aluminum alloy which we use for our training set does not seem expedient. We choose to train the model to predict a "relative density" which rather represents the degree of filling at given position in the CT scan. We assign a relative density of 1.0 to voxels which are completely filled with material. In contrast, voxels which are completely part of a defect or the surrounding air are assigned a relative density of 0.0. The occurrences of structural loosening have any gradation in between and inclusions of foreign material have any value greater than 1.0. Figure 4.17 shows an exemplary patch of a CT scan and its according relative density map. We expect the values of the result to be in the range $[0, \infty[$. Hence, we use the linear output of the final layer directly.

Again, we could use the MSE loss to train the model. However, the MSE loss is vulnerable to outliers and, therefore, to exploding gradients because the loss grows quadratic with the distance to the optimal value. Less sensitive to outliers would be the mean absolute error (MAE) loss as it has a constant gradient independent of the distance to the optimal value. This is its biggest weakness, though: as the gradient magnitude is the same even if the value is close to a local optimum, which makes it easy to miss the optimal value. The combination of both losses is called Huber loss [92]. For small values, it switches from the MAE loss to the MSE loss, whereby "small values" are defined in accordance to a new hyper-parameter δ. Another loss function which has the same benefits as the Huber loss but is two times derivable everywhere (as it contains no abrupt changes) is the log-cosh



(a) realistically simulated CT scan (b) relative densities

Figure 4.17.: Training data for the density regressor: We train the density regressor with a relative density. Our aluminum alloy which has an actual density of 2.80 g cm^{-3} is assigned the value 1.0. Defects and the surrounding air have the value 0.0, the structural loosening have a density closer to 1.0 and inclusions of foreign material a density greater than 1.0. Voxels which lie on the boundary of two materials are assigned an interpolated value.

loss [198] (see Equation (4.3)). This further means there is no additional hyper-parameter which needs to be tuned.

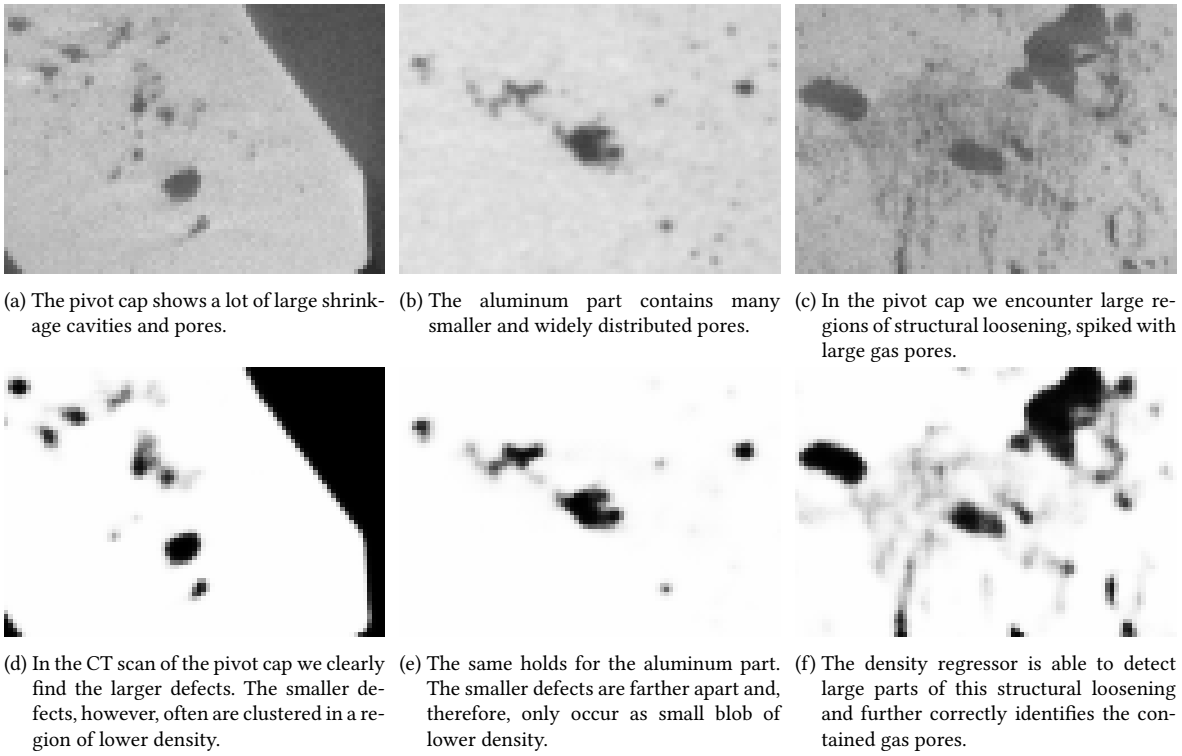
$$\mathcal{L}_{\text{log-cosh}} = \log(\cosh(y - p)) \tag{4.3}$$

Figure 4.18 shows some density maps generated by the density regression model. In contrast to the input CT scans the boundaries of the defects seem to be less crisp. This also affects smaller instances which never reach a density value of 0.0 even though they are larger than two voxel in diameter. Additionally, we are able to detect occurrences of structural loosening (see Figure 4.18f). Yet, we cannot fully prove that the predicted relative density matches the actual density of the structural loosening.

Despite the results looking very promising, it is hard to obtain a precise segmentation from the density maps as the densities are continuous values and do not allow us to draw a globally optimal threshold. Nevertheless, the density map could be used as secondary output providing further information for a subsequent decision about the severity of a defect.

Semantic Segmentation

In the end, we need a binary segmentation, so we should train our model to directly output a segmentation mask by assigning a label to each voxel. This approach corresponds to the traditional machine learning model, but instead of using hand-designed features and a downstream classifier we combine both in a single FCN. The labels for this approach are one-hot-encoded class assignments, giving each voxel a certain class. This means we have to draw a hard threshold on our ground truth labels. Depending on where we draw the threshold we affect the sensitivity of the model. For example, if only 10 % of a voxel need to be part of a defect to be labeled as defective, the trained segmentation model will be more sensitive than by using a higher threshold to binarize the ground



(a) The pivot cap shows a lot of large shrinkage cavities and pores.

(b) The aluminum part contains many smaller and widely distributed pores.

(c) In the pivot cap we encounter large regions of structural loosening, spiked with large gas pores.

(d) In the CT scan of the pivot cap we clearly find the larger defects. The smaller defects, however, often are clustered in a region of lower density.

(e) The same holds for the aluminum part. The smaller defects are farther apart and, therefore, only occur as small blob of lower density.

(f) The density regressor is able to detect large parts of this structural loosening and further correctly identifies the contained gas pores.

Figure 4.18.: The results of the density regressor are very promising to provide further information about the part under examination. However, they are not helpful for finding a precise segmentation of individual defect instances.

truth. We expect the output to be a probability distribution over the possible classes. Therefore, we compute a softmax [71] over the output channels of the final layer.

Normally, a segmentation model could be trained using cross entropy loss [71]. The cross entropy loss measures how well the predicted values match the ground truth taking true negative predictions into account, too. Due to the highly imbalanced nature of our training set this would tremendously increase the convergence time of the model. Hence, we need a loss function which puts the focus on the target classes, i. e. the dice loss [19, 199, 200]. To further control the sensitivity of the trained model, we switch to a generalized version: The Teversky loss [184, 201] introduces two parameters (α and β) which allow us to control the weight of false positive and false negative predictions (see Equation (4.4)).

$$\mathcal{L}_{\text{Tversky}} = 1 - \frac{TP}{TP + \alpha \cdot FN + \beta \cdot FP} \tag{4.4}$$

This enables us to train a high recall encoder-decoder-pair by choosing $\alpha = 1.5$ and $\beta = 0.5$ for the intermediate layers. For the refinement-step, we then use a balanced dice loss ($\alpha = 1.0$ and $\beta = 1.0$) to achieve a higher precision for the final segmentation output. Some qualitative results of the segmentation model are shown in Figure 3.11. With this model, we obtain the precise segmentation which we need to extract further properties of the defects to provide sufficient information for a proper subsequent assessment. This method will further be referred to as “deep learning method”.

4.3.3. The Training Process

We train our model for 150 000 iterations using an Adam optimizer [72] with an initial learning rate of 10^{-4} and $\beta_1 = 0.9$ and $\beta_2 = 0.999$. With the initial learning rate we differ by a factor of 0.1 from the recommendations which are described in the paper because we encounter exploding gradients with higher learning rates. The benefit of the Adam algorithm is that it maintains an individual learning rate for each parameter of the neural network and adapts it according to estimations of the first and second moment information of the gradient.

For training, we crop patches of $128 \times 128 \times 128$ voxel from our CT scans (which have a size of $1000 \times 1000 \times 1000$ voxel). Loading an entire CT scan to pick a small patch and then discarding the rest would introduce a huge I/O overhead and considerably slow down the training process. Therefore, we maintain a *cache* of ten entire CT scans in memory. For each patch, we randomly pick a CT scan from the cache, extract the patch and put it back. A background task, the *cache keeper*, takes care to continuously exchange the CT scans in the cache so that we iterate through the complete training set. Another approach would be to extract the training patches before starting the training process. However, this would at least duplicate the amount of data we have to store on our hard drives and we would be limited by the bandwidth to the hard drive again instead of a much faster connection to the main memory. Moreover, having the entire CT scans in the cache at hand, allows us to arbitrarily increase the variety of patches by randomly shifting the patches around given positions in the CT scan. The CT scans are stored as 16 bit unsigned integer values. Hence, we need to normalize the values to the range $[0, 1]$ before feeding the data to our model. We do this by subtracting the minimal possible value and dividing by the maximal possible value, i. e. 0 and 65 535, assuming the CT scan utilizes the full data range. This has to be considered during the inference, too, and the data has to be normalized accordingly. Training our model on a NVIDIA TITAN RTX using a batch size of two, it takes about three days for the model to converge. Training the standard U-Net with a batch size of five is much faster and only needs about a day. Using the full three-dimensional context not only comes with an increased computation time at inference time but also impacts the training process.

To prevent our model from overfitting to the training data and to further increase the variety of the data, we use *data augmentation* [58, 71]. The basic augmentation operations comprise the rotation and mirroring of the input as well as randomly cropping smaller regions of the training images. The latter is inherent to our training method, because we pick our training patches as parts of larger CT scans. Changes in the color are not possible for a single channel input, however, we change contrast and brightness of the training input and perform a gamma correction on the input data. This not only prevents the model from overfitting but also makes it more robust against changes in the target data distribution. The positions of the air and material peaks in the histograms of the CT scans sometimes widely differ from those of our training set. This happens when the acquisition parameters do not adhere to the assumptions we make for our training set. While these data augmentation techniques are pretty similar to those of the domain randomization (see Section 3.2.1), we are more restrictive with the selection of the parameters to stay within more meaningful boundaries. In addition, we explicitly do *not* superimpose structures into the image which resemble artifacts like gray-value gradients (cupping artifacts), rings (ring artifacts), or white noise (Poisson noise). They are contradictory to the artifacts introduced by the realistic simulation. The augmentations are computed on the fly from randomly chosen parameters. Figure 4.19 shows a few examples of augmented patches. Another step we take against overfitting is adding a regularization loss to the weights pushing them

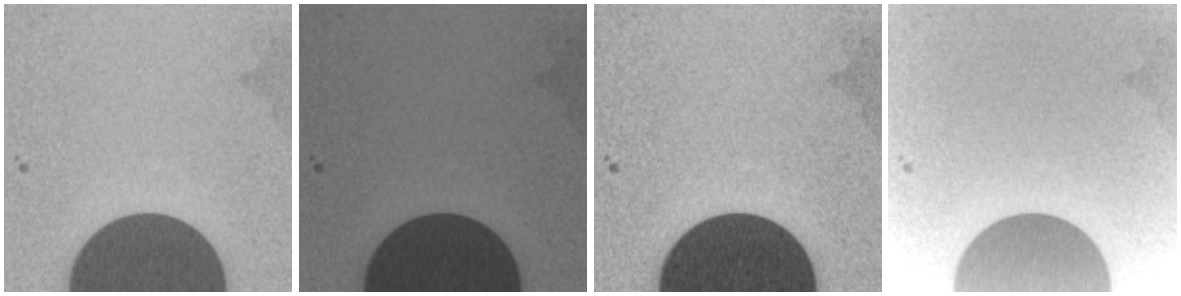


Figure 4.19.: With well-dosed data augmentation we modify our training data to prevent our model from overfitting and to make it more robust against variations in the scan process. For example, we change the brightness and contrast of the input image and perform gamma corrections with randomly chosen parameters. Some examples are shown in the first three images from left. The right most image shows a superimposition of a gray-value gradient, which we do not include in the final set of data augmentations because it produces patches which show artifacts that are contradictory to those of the simulations.

slowly towards zero when not receiving any updates. For this we use a L_2 regularization [71] term and weight it by $1e^{-4}$.

Because of the highly imbalanced nature of our data, we decide to apply a curriculum learning [202] strategy with fixed steps to provide our model with sufficient information about its target class, especially in the early iterations. The learning schedule starts with patches which are randomly chosen from defect positions, then we slowly add more and more patches showing the material boundary and, finally, we add patches which are randomly chosen from the entire CT scan. To support data augmentation we do not always use the same position for defect but rather randomly shift the center of the patch by ±16 voxels along each axis. Another aspect of the curriculum is to slightly increase the amount of data augmentation as the training proceeds. We start using only rotation and mirroring augmentations (and the translation we obtain from randomly shifting the position from which we crop our patch). We hope that an easy start allows the model to get an initial idea of its task and to form first features and, therefore, to converge faster. After 25 000 iterations, we add adjustments in brightness and contrast and a gamma correction to the list of augmenters. Furthermore, our caching strategy contributes to the soft start of the training process, by not exposing the full variety of our realistically simulated training set to the model at once. The patches for the first few iterations are all chosen from the same CT scan. Then, the variety is increased little by little by adding more and more CT scans to the cache and then by constantly replacing the CT scans.

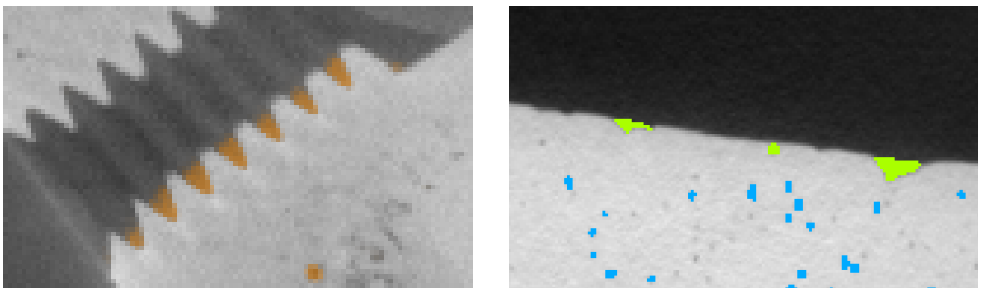
We use TensorFlow (Google Brain) to model and train our deep neural networks. There are other equally powerful frameworks like PyTorch (Facebook) or Apache MXNet. However, when we started working on defect detection, TensorFlow was the only framework to support Microsoft Windows as operating system, which is a huge advantage when developing a product for the industry. Because of TensorFlow being already adopted by a big community, there are plenty of examples and guides, which is helpful when encountering any technical issues. Moreover, most of the research papers published at this point in time provide exemplary code which uses TensorFlow. Unfortunately, their API is not set in stone and when switching versions there is a good chance to encounter unforeseen difficulties.

4.3.4. Tuning the Model Output

Having the model architecture, the target function, and the training method at hand, it is time to further tune the output of the deep learning method. We notice that the original model which is naively trained to distinguish between "defective" and "flawless" has a hard time with structures in the boundary region of the scanned part. Especially screw threads are a challenging formation (see Figure 4.20a). This might be because our initial training set includes surface defects, i. e. defects which penetrate the material boundary of a part, but only contains flat material boundaries.

Explicitly including screw threads in the training data, however, is not sufficient. Therefore, we separate surface defects and inner defects in two distinct classes. The trained segmentation model, now, not only is able to tell apart inner defects from those which penetrate the material boundaries of the part under examination (see Figure 4.20b), but the false positive responses in small surface structures like screw threads become less significant. This means we do not merely shift the false positive responses from one class to another but instead enable the model to yield a more precise segmentation by providing further information. We can provide even more information by further training with the segmentation of occurrences of structural loosening. Even though the results indicate where an instance of structural loosening occurs in the part (see Figure 4.21), it is far from a segmentation which is as precise as the one of the defects.

What remains to test is whether the model becomes even better and more robust if we provide further guidance during training, e. g. by utilizing meta information of the scan and reconstruction processes. For the proper avoidance of labeling ring artifacts as defective, it could be beneficial to let the model additionally predict for each voxel its distance to the rotation axis. The distance to the material boundary as additional target could (with reservations) be helpful for dealing with artifacts which result from beam hardening. Another additional output could be the material boundary of the part, excluding all defects. More difficult to include is the direct prediction of derived properties of defects like their volume or sphericity as they concern defect instances and cannot be broken down into voxel-wise labels.



(a) false positive responses in screw threads

(b) separation of surface defects and inner defects

Figure 4.20.: A segmentation model which is trained to simply tell apart defective and non-defective voxels, i. e. a two-class model, often mistakes structures on the surface of the part for defects, for example, screw threads (a). Moving the surface defects to a separate class not only allows us to distinguish between surface defects and inner defects (b), which is beneficial because cavities on the surface of a part might be more critical, it further reduces the significance of the false positive responses by far. The orange overlay in (a) shows the defect probability output of the model which was trained to predict inner and surface defects jointly. In (b) the predicted inner defects are shown in blue and the separately predicted surface defects in green.

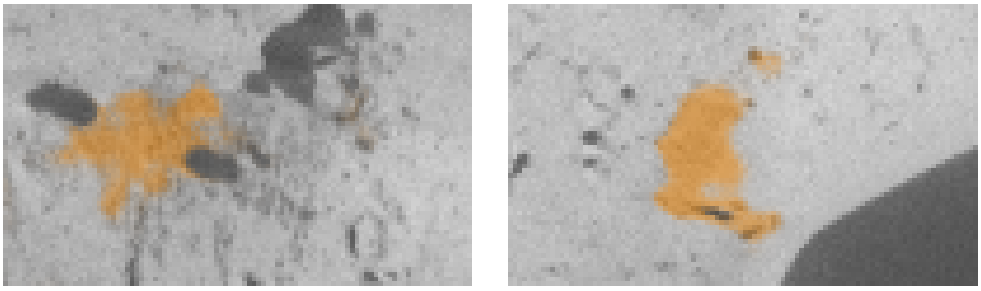


Figure 4.21.: We provide even more information to the model by adding more classes to the training process, e. g. structural loosening. However, the results are not as convincing as we would expect them to be, yet. The orange overlay shows the prediction output for instances of structural loosening.

Even tough the FCN architecture would allow us to process arbitrarily large inputs, we need to process the CT scans block-wise due to the limited amount of available GPU memory. Despite being trained to yield results which are invariant to rotation, translation, and scale it can happen that there is a clearly visible cut in the result when simply stitching together the block-wise outputs. This is due to the missing context information in the border region of each block. Hence, we process a block as large as the GPU memory allows us to, but we only use the inner part of the output for the final result. The more overlap the blocks have, the more homogeneous the results become. We use an overlap of 32 voxels in all dimensions. This increases the necessary computation time as we need to process parts of the volume up to eight times but it also significantly improves the result. (This overlap is already included in the computation time of about 500 s – 540 s we mention before, using blocks of $320 \times 320 \times 320$ voxel.)

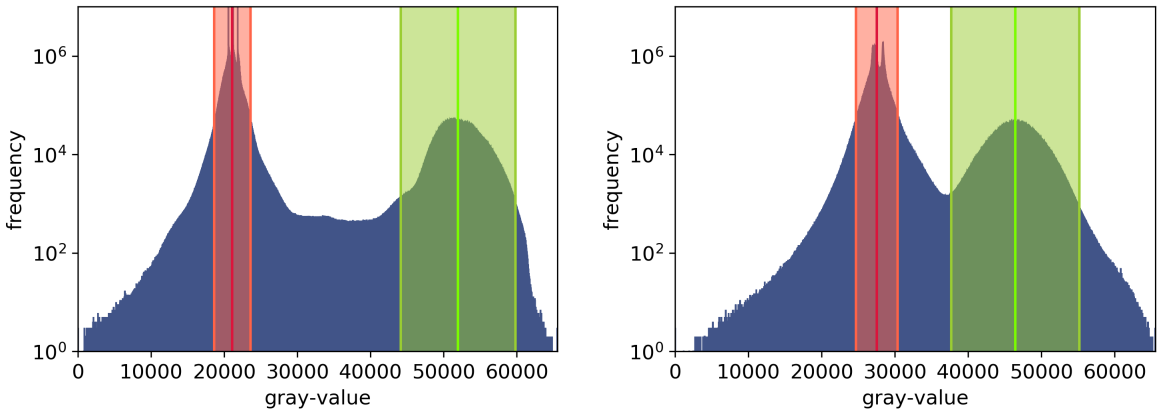
After exploring and discussing several methods for the reference-free detection of defects, we now have a deep learning approach and additionally two promising traditional methods to compare with: (i) an image processing method which makes no use of machine learning at all, (ii) a voxel-classification traditional machine learning pipeline using a random forest, and (iii) an end-to-end trainable deep learning model. In the next chapter (Chapter 5) we dive deeper into the evaluation techniques and in Chapter 6 we further explore the suitability of those models for more specific real-world scenarios.

5. Evaluation of Highly Imbalanced Data

Usually we expect the majority of a cast aluminum part to be flawless. This is reflected by our validation set: Each CT scan consists of $1000 \times 1000 \times 1000$ voxel and only few of them are part of our target class, i. e. only 0.021 % of the data is defective. We, therefore, have to deal with highly imbalanced data not only during training but also when validating our models. Because the fraction of "defect"-voxels is so low, a model which labels each and every voxel as "flawless" would already achieve an accuracy of 99.979 %. This would by no means be useful. Adding more defects to balance the data would increase the "reality gap": If we had 50 % of the voxels as part of defects, we would reduce the penetration length which again would result in an unintended high scan quality. Moreover, only about 20 % of the CT scan show the actual cast aluminum part. The rest is either air or outside the cylindrical field of view. Neither can we increase the fraction of the cast aluminum part in the CT scan as it would reach outside the field of view. Hence, we have to turn to measures which neglect the prediction of true negatives and put more focus on the target class. We evaluate two measures: (i) the probability of detection (POD) which tells us which fraction of the defects of a given size can be found with a given confidence in a CT scan of given quality [203, 204] (see Section 5.2) and (ii) the intersection over union (IoU) which tells us more about the overall quality of the segmentation output [205, 206] (see Section 5.3). While the POD is well known to the domain experts [204, 207–209], the IoU is the standard tool for evaluating imbalanced data in the machine learning world [206]. Furthermore, in Section 5.4 we provide a precision recall curve (PR curve) [210–212], which is the equivalent of the receiver operating characteristics (ROC) [92, 213] for highly imbalanced data to find a good operating point. But first of all, in Section 5.1, we analyze the realistically simulated training data to put the evaluation results into context.

5.1. Statistics of The Training Set

In order to better interpret the results, it is important to get a notion of the structure of the training and validation set. If we know how the data is composed, we know what we can expect from the results. First, we have a look at the simulated CT data. A rough measure of the *data quality* of a CT scan is its CNR. The CNR is defined as the difference of material peak and air peak divided by the width of these peaks, i. e. $\mathrm{CNR} = (\mu_{\mathrm{material}} - \mu_{\mathrm{air}})/\sqrt{\sigma^2_{\mathrm{air}} \cdot \sigma^2_{\mathrm{material}}}$ [33, 34, 214]. Alternatively, we can compute it with the help of a special test specimen, as defined in Section 5.2.2. The CT scans with the worst data quality in our data set have a CNR of 6.6, the ones with the best data quality a CNR of 12.5. With that we are able to cover a wide range of data qualities during training and validation. In addition, we take a closer look at the histograms of our CT scans. The CT scans of our data sets utilize the full data range of 16 bit unsigned integers which allows for high gray-value dynamics. The air and material peak of CT scans of good data quality are well defined and far apart (see Figure 5.1a). For the CT scans of worse data quality the air and material peak are wider and closer together (see Figure 5.1b). Of course, this is also expressed in the CNR, however, we need to



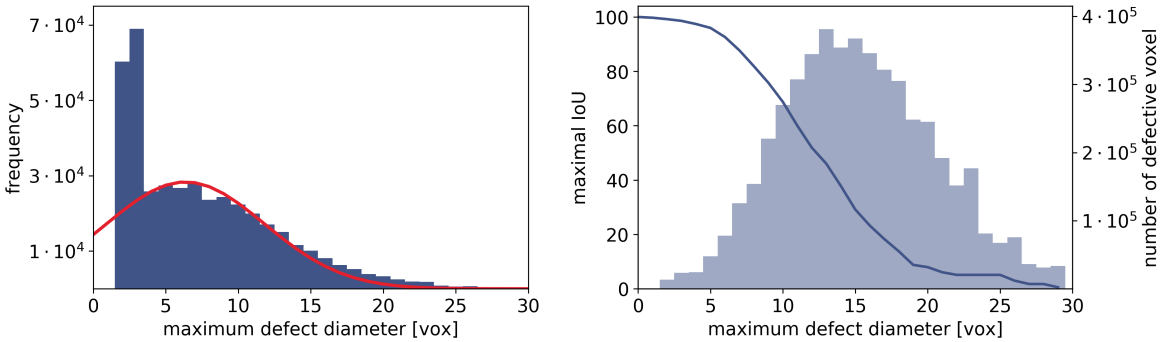
(a) A histogram of a CT scans of good data quality.

(b) A histogram of a CT scans of bad data quality.

Figure 5.1.: The CT scans in our training data are all stored as 16 bit unsigned integer and utilize the full data range. The material peak (green) and the air peak (red) in the histograms of the data, however, vary depending on the data quality. For CT scans of good data quality we have a high contrast and the peaks are far apart (a), while for CT scans of bad data quality the peaks are close together (b). Furthermore, if the data is more noisy, the peaks are wider.

consider the position of the air peak and the material peak also for the inference on unseen data. Our models are trained with floating point values in the range [0.0, 1.0]. The 16 bit unsigned integer values of the CT scans are mapped to [0.0, 1.0] so that 0 is mapped to 0.0 and the maximum possible value of 16 bit unsigned integers, i. e. 65535, is mapped to 1.0. Therefore, during training, the air peak is somewhere between 0.31 and 0.43 and the material peak is somewhere between 0.67 and 0.81. In production, however, the histograms may vary, for example, if they do not fully utilize the data range. Hence, it is important to map the gray-values of the data to the range [0.0, 1.0] in a way that the air and material peaks approximate those of our training set, i. e. the new input matches the data domain of our training data. By augmenting contrast and brightness during training, we mitigate the impact of an improper mapping. Nevertheless, we achieve better results using a well defined mapping.

Next, we investigate our target class, the *inner defects*. In total we have 418 095 defects in the training set. However, while the training set comprises 675.00 billion voxels and of those about 94.40 billion voxels are part of our virtual cast parts, only 0.14 billion voxels actually show defects. More important than the mere number of defects is the distribution of their size and how this affects measuring the IoU. The diameters of our defects are chosen from a normal distribution with a mean of 6.36 voxels. Due to the clamping of values which are smaller than 1.82 voxels, we have more than twice as many tiny defect instances than we have instances of other sizes (see Figure 5.2a). For the computation of the IoU, the number of voxels matters. Therefore, we calculate the maximum IoU which is achievable when not detecting defects smaller than a given diameter. Despite having a lot more tiny defects (with a diameter smaller than 5 voxels), their voxel count carries almost no weight when computing the IoU. We observe the highest significance for defects with a diameter between 5 voxels and 18 voxels (see Figure 5.2b for a more detailed correlation of the IoU and the voxel count). This, in addition, means that by not detecting any defects smaller than 5 voxels diameter, it still is possible to score an IoU of 97.4 %. However, for larger defects the impact of a slightly imprecise segmentation becomes much more significant. The same gray-value and defect distributions hold



(a) The histogram of the number of defects as a function of their maximum diameter in dark blue. In red we plot the corresponding normal distribution used in the simulation pipeline to pick the defect sizes.

(b) The maximum IoU which is still achievable when not detecting any defects which are smaller than a given minimum size is shown in dark blue (considering the maximum diameter of each defect). The corresponding total amount of voxels for defects with the given size is shown in light blue.

Figure 5.2.: The peaks in the number of defects with a maximum diameter of 2 voxels and 3 voxels occur due to the clamping of the defect size to at least 0.2 mm when creating the defect meshes (a). Despite their large number, the defects which have a diameter smaller than 5 voxels hardly influence the IoU (b).

for the validation set—only the absolute values are reduced by about a factor of 12.5.

The distribution of the *surface defects* is similar to the distribution of the inner defects. However, it has a larger standard deviation and the minimum size is limited to 9.1 voxels. Smaller surface defects are hard to distinguish from a rough surface. Initially, we place 33 750 defects, but sometimes they are placed within screw threads and span across multiple threads. Therefore, the defects are split into multiple smaller instances and we end up with 35 154 surface defects. This further means that we have some instances in the data set that are smaller than the initially defined minimum size. The surface defects make up 0.06 billion voxels in our data set. With the statistics of our data set in mind, we now turn to the evaluation of our reference-free defect detection methods.

5.2. Probability of Detection

The POD is an instance-based measure telling us the probability at which a defect of a given size can be found in a CT scan of a given data quality. This measure is widely used among domain experts. Therefore, evaluating our algorithms in terms of the POD helps to further build up trust among the user group. However, there are different ways to obtain this measure [203, 204]: The first method is the *"hit/miss method"*. This method requires a clear definition of a defect to be counted as found, for example, a minimum confidence of 95 %. The confidence is usually defined as the fraction of correctly found voxels of an instance. For each defect instance, we determine if it is properly detected in accordance with the given criterion, i. e. a *hit*, or not, i. e. a *miss*. The POD for a given defect size (or cluster of defect sizes) is then computed as $\mathrm{POD} = n_{\mathrm{hit}}/(n_{\mathrm{hit}} + n_{\mathrm{miss}})$, where n_{hit} is the number of detected instances of a given defect size and n_{miss} the number of instances of a given size which is not found. The second method is the *"â vs. a method"*. This method is a non-binary method which for each instance takes the confidence into account at which it is detected [203]. The confidence again is defined as fraction of correctly found defect voxels. In Figure 5.3a we compare

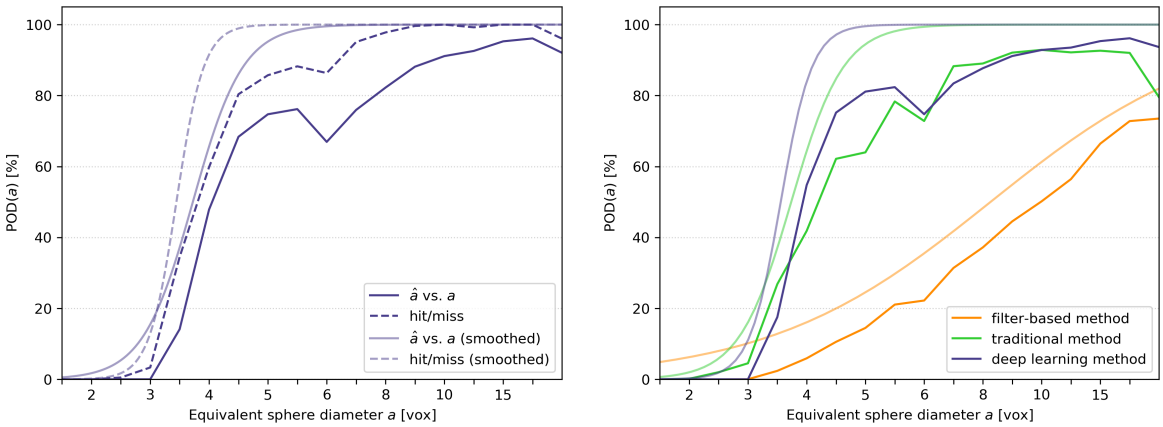
both methods. We observe that for the same data, the hit/miss method tends to lead us to believe in the methods being better than they actually are. This is because the hit/miss method counts any defect instance as hit which is above the confidence criterion independently of whether it is only slightly above or at 100 % confidence. The $\hat{a}$ vs. a method in contrast further considers the difference to 100 % confidence. Therefore, we choose the $\hat{a}$ vs. a method as it contains more information. The following sections describe how to compute the "$\hat{a}$ vs. a method" (see Section 5.2.1) and how to put it in context with the data quality measurements of the CT system (see Section 5.2.2).

5.2.1. How to Compute the Probability of Detection

Computing the POD following [203] involves a lot of smoothing and re-sampling but yields more interpretable results. First of all, we need to binarize our segmentation mask. As the POD itself does not take false positives into account, choosing a threshold which labels everything as defective would yield the best results. However, this would in no way be meaningful. Hence, we have to make sure to keep the false positive predictions as low as possible and evaluate the POD for the threshold which achieves the best IoU (see Section 5.3). For each defect instance in the ground truth, we compute the *equivalent sphere diameter (ESD)*, i. e. the diameter which a sphere of the same volume as the defect would have, and the detection confidence a, i. e. how many voxels of the defect are actually predicted as "defective". Then, we *cluster the defects* by their ESD and compute the mean μ_c and standard deviation σ_c of the confidence values for each cluster c. This has the possibility to render some quantization artifacts into the data as smaller defects only have fewer voxels which can be found. μ_c and σ_c are used to perform a *bootstrap re-sampling*, drawing 1000 new samples from a normal distribution $\mathcal{N}(\mu_c, \sigma_c)$. This way, we have the same number of samples in all clusters. All samples which are smaller than a given minimum confidence $\hat{a}$ are discarded, i. e. set to 0. Finally, the POD for a cluster c is computed as the mean of these samples (Equation (5.1)). We plot the POD as a function of c. Normally, the resulting curve would be smoothed further by performing a *logistic regression*, which we decide to skip to not further manipulate the results and shift the values towards smaller defect sizes (see Figure 5.3a).

$$\mathrm{POD}_{\hat{a}}(c) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} a & \text{if } a \geq \hat{a}, \\ 0 & \text{otherwise} \end{cases}, \quad \text{with } a = \mathcal{N}(\mu_c, \sigma_c) \tag{5.1}$$

In Figure 5.3b we show the mean POD which our reference-free defect detection algorithms achieve on our validation set. The results comprise all the CT scans of all data qualities. We set the confidence threshold to 80 %. We see that, as expected, the machine learning-based methods achieve significantly better results than the plain filter-based method. With the use of machine learning we detect defects with a ESD of only 4 voxels at 50 % probability. To achieve the same with the filter-based method we need to double the ESD. While the traditional method shows slight indications for defects with an ESD smaller than 3 voxels, the deep learning method only starts to show indications for defects with an ESD larger than 3.5 voxels. However, the deep learning method starts off with a much higher probability, outperforming the traditional method. As we compute the POD thresholding the prediction at the threshold that maximizes the IoU in each case there is a good chance for false positive responses. Therefore, we further report the precision (measured per voxel) of our methods. Usually, the false positive rate (or possibility of false alarms) is reported. However, this



(a) When comparing the hit/miss method with the $\hat{a}$ vs. a method, we see that the hit/miss method tends to yield better results, while the $\hat{a}$ vs. a method is more conservative. In addition, we see that by fitting a sigmoid to the results shifts the curve towards better values.

(b) When computing the mean POD over all CT scans in the validation set for our three reference-free defect detection methods, we see that the machine learning methods detect smaller defects more confidently than the filter-based method. The lighter plots show the respective smoothed POD.

Figure 5.3.: We compare the two different methods to compute the probability of detection (a) and compare our three defect detection methods in terms of the probability of detection using the $\hat{a}$ vs. a method (b). These results are computed using the data of the validation set.

measure again includes the true negatives and, hence, would always be almost zero. For the filter-based method, the traditional method, and the deep learning method we get a precision of 91.0 %, 91.7 %, and 92.4 %. The deep learning method yields the least false positive responses. A drawback of using the ESD for this evaluation is that elongated defects which in general are much harder to detect than roundish defects of the same ESD are put into the same cluster. If we now have much more roundish than elongated defects in a cluster of defects with a smaller ESD and in the next cluster we have more elongated defects, this can distort the results. This explains the dip in the POD for a ESD of 6 voxels in Figure 5.3b.

A characteristic value of the POD is the $a_{90/95}$ value at which a defect of given size a can be found with a probability of 90 % at a confidence level of 95 %. For this, the POD needs to be computed with a confidence threshold of 95 %. The domain experts define this value as the minimum defect size which allows a trustful and reliable detection [207]. However, using the smoothed POD curve for this value has a huge impact on the $a_{90/95}$ value. For our deep learning method we achieve an $a_{90/95}$ value of 8.3 voxels on our validation set using the smoothed POD. For the traditional method we obtain an $a_{90/95}$ value of 9.3 voxels. For the filter-based method we could not compute a $a_{90/95}$ value because only for the CT scan with the best data quality the filter-based method reaches a POD of 90 % at a confidence level of 95 % at all.

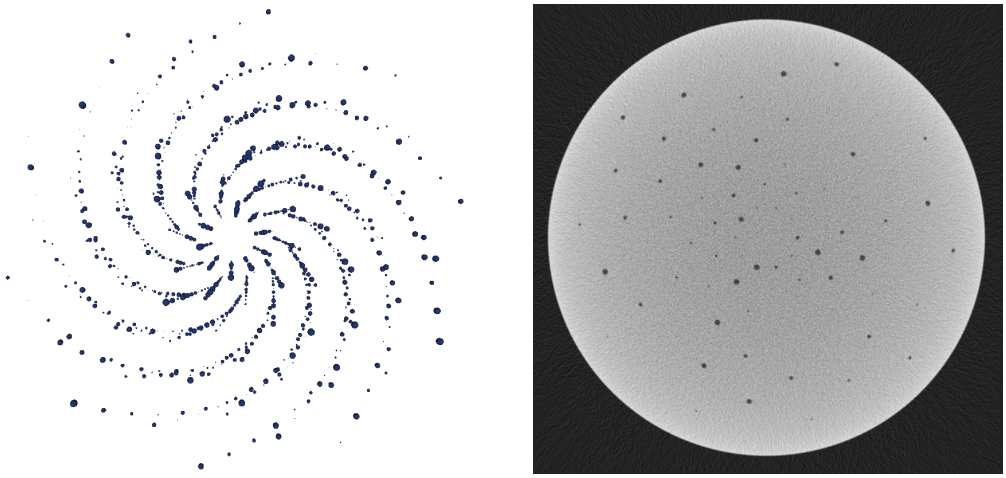
5.2.2. A Special Test Specimen

A very flat crack with a large extent might be far less visible than a spherical gas pore of the exact same volume. The crack might only be one or two voxels thick, gaining its volume from its large extent. In contrast, the gas pore has about the same diameter in every dimension. This distorts the

POD: We obtain a good detection for the gas bubble, while the detection confidence of the flat crack might be considerably worse. For the computation of the POD to be independent of the actual shapes of the defects and, therefore, to be more comparable, we design a special test specimen and use our simulation pipeline to create realistic CT scans. This test specimen is inspired by the standards which define how to measure the performance of a CT system (ASTM E 1441 [215] and ASTM E 1695 [216]). Thus, we define the specimen in accordance with those standards: The test specimen has to be a cylinder of the same material and about the same penetration length as the actual target part. Then, we place 880 spherical defects, distributed over 11 discs with 8 rings each throughout the cylinder (see Figure 5.4a). The size of the defects has to range from barely detectable to clearly visible. This depends on the spatial resolution of the CT system, i. e. the size of the voxels and the focal spot size. We suggest that the smallest defects should have a diameter of about two voxels, while the largest defects should be bigger than two transitions from material to background plus some margin, in our case about 10 voxels. For a smooth plot, there should be at least ten gradations in size. Each size has to be present in each ring. This allows further evaluations of the detection rate as function of the distance to the material boundary, i. e. the influence of beam hardening on the detection. (We will analyze this in Section 5.3.2 in terms of the IoU in our artifact-space.) The sizes of the defects need to be drawn randomly (without returning) for each ring from the possible sizes so that the different sizes are distributed equally over the possible positions. With a regular pattern strange quantization artifacts occur, causing an early spike in prediction performance for certain defect sizes even though the IoU is about the same for the regular pattern as for the random pattern. This is because we compute the POD on voxel-basis, while the actual defects are continuous. Therefore, quantization artifacts and the binarization of ground truth and prediction mask can add up beneficially for defects of a certain size, if these instances are always aligned in the same way within the voxel grid. This can cause the spikes which we observe in the plot (see Figure 5.5a). Additionally, the rings of defects need to be slightly rotated against another to maintain equal penetration lengths and to avoid further build ups of impeding effects. With all defects having one of n different sizes, we do not need to cluster the defects by their ESD. Instead, we use the diameter of the meshes representing the defects. Figure 5.4a shows the arrangement of defects with randomly chosen defect sizes. The defects in our specimen range from 100 µm to 1000 µm which corresponds to 0.91 voxels and 9.09 voxels, respectively. We use ten different sizes ($n = 10$). The cylinder, in accordance with the training data, has a diameter of 80 mm and a height of 45 mm. We scan the test specimen with the same parameters which we use for our training set (see Figure 5.4b).

As we design our POD specimen in accordance with the standards for evaluating a CT system, we are able to combine the POD with the information we obtain from the data quality measures which are defined in these standards, namely the modulation transfer function (MTF), the contrast discrimination function (CDF), and the contrast detail dose (CDD).

Modulation transfer function Based on the material boundary of the test specimen, the MTF quantifies the spatial resolution. To mitigate the disturbances of image noise, we first bin the gray-values of all voxels which lie within a certain margin by their distance to the material boundary. We so obtain the edge response function, i. e. the transition from material (or foreground) to air (or background). Its derivative is the point spread function. The peak of the point spread function which is centered around the edge encodes the information of how wide the edge is. Using the Fourier transformation we convert the point spread function to the MTF. For more implementation details, we refer to the standard ASTM E 1695 [216]. In order



(a) Defect model of the test specimen for a more comparable POD calculation.

(b) A slice of the realistically simulated CT scan of the test specimen.

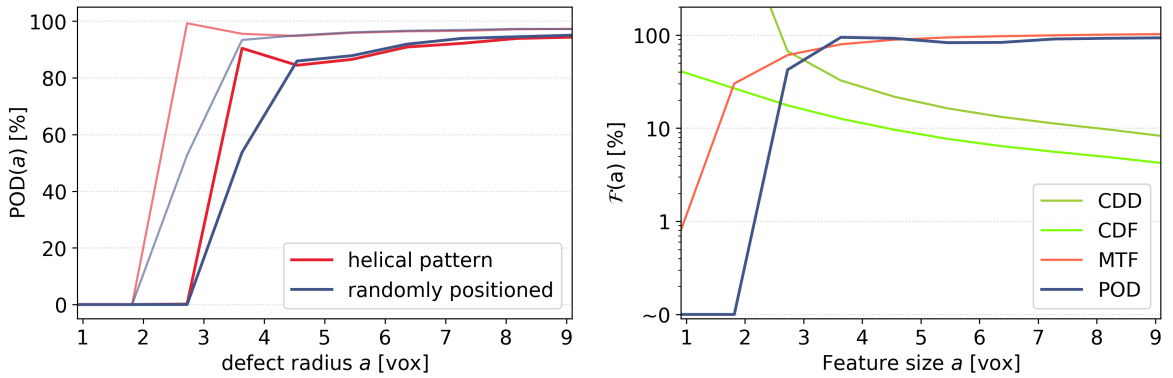
Figure 5.4.: For a comparable POD calculation, we create a special test specimen with the defects arranged in concentric circles. All defect sizes are distributed randomly within each ring to avoid discretization artifacts in the final plot. This specimen further allows for the computation of data quality measurements like MTF, CDF, and CDD in accordance with the standards ASTM E 1441 [215] and ASTM E 1695 [216].

to compare the results to those of the POD and for the computation of the CDD, we need to transfer the MTF back into the spatial domain, sampling the MTF at the desired feature sizes (compare ASTM E 1441 [215]).

Contrast discrimination function Based on the inner part of the test specimen, the CDF quantifies the contrast sensitivity. To compute the CDF an iteratively increasing tile pattern is used. In each tile we compute the mean gray-value. Then, we compute the standard error in the tile set and increase the tile size. The CDF, therefore, shows the standard errors as a function of the tile sizes. For more implementation details, we refer to the standard ASTM E 1695 [216]. From the CDF value with a tile size of 1, we can further compute the CNR by 3/CDF(1) [216].

Contrast detail dose The CDD combines the MTF and the CDF with a so called physiological factor. This factor should reflect the capability of the auditor, trying to quantify what an auditor is able to see [215]. The CDD is computed as the physiological factor times the CDF divided by the MTF (in the spatial domain). For more implementation details and more information on the physiological factor we refer to the standard ASTM E 1441 [215].

Originally, all these measures are calculated using a single slice of the data taken from the center of the CT scan. However, in our case the CDF cannot be computed in the central slice because the inner region, which is defined by $\frac{1}{3}\,r$, is filled with defects which would disturb the calculation. We deliberately place the defects in the center slice to further obtain POD values for the best reconstruction, i. e. the region with the least Feldkamp artifacts. Consequently, we compute the quality measures in a slice centered between the central disk of defects and the next disc of defects, assuming that the Feldkamp artifacts from the reconstruction do not carry as much weight at that position. With the POD we can tell for a given system and a given CNR which defects can be recognized at which confidence. We can now change our CT system, altering the CNR of the CT scans in order to de-



(a) If the defects align with the voxel grid in an unfortunate way, discretization artifacts manifest as a strange spike in the plot. Randomly choosing the defect sizes mitigates the discretization artifacts, resulting in a smooth plot.

(b) Using the test specimen, the results of the POD can be easily plotted side by side with the data quality measures defined in [215,216]. Note that the y-axis is in logarithmic scale as required by [215]. The CNR of the CT scan is 7.4.

Figure 5.5.: The test specimen needs to be designed properly to avoid discretization artifacts (a), but then it allows for a comparison with the data quality measures in accordance with the ASTM standards (b).

termine how robust our defect detection method is. The idea behind comparing the POD to these data quality measures is to enable the users of a defect detection system to scan a solid cylinder (without defects) in accordance with the standards [215, 216] and to, then, draw conclusions about the results they can expect. However, correlating these measures is difficult. Even though it seems to be a good idea to compare the MTF to the POD as the MTF measures one transition from bright to dark, and the POD basically measure two transitions (white to dark to white) there is a significant difference between both. The MTF is measured at the outer material boundary, which only slightly changes when decreasing the data quality. The POD, in contrast, is measured at tiny defects in the center of the material which tremendously change their appearance when decreasing the data quality. The CDF is also measured at the center part of the test specimen, however, it only carries information about the image noise in the CT scan. The CDD which is a combination of MTF and CDF could, therefore, be suited to draw conclusions about the detectability of defects. But it includes an arbitrarily chosen factor, the physiological factor. Nevertheless, for a physiological factor of 2.0 we obtain a quite good match of the CDD-50 and the POD-50 value, i. e. the feature size for which both curves cross the 50 % line. For the deep learning method we obtain a factor of about 1.45, for the traditional method a factor of about 1.56, and for the filter-based method a factor of about 1.93. While this is only a rough correlation and does not allow to provide any guarantee, it could give a good indication of how well the system will work.

In Figure 5.6 we show the corresponding POD plots for the filter-based method, the traditional method, and the deep learning method. We evaluate three different data qualities which have a CNR of 14.8, 9.8, and 4.8, respectively. For the deep learning method, we see that it does not detect any defects with a radius smaller than 2 voxels. After that, the detection probability quickly grows towards 100 %. In addition, for the deep learning method, the POD curves of all data qualities lie close together, which indicates a high robustness against changes in the data quality. However, we observe a drop at larger defects. This probably is due to a slight under-segmentation of larger defect instances. With the POD we observe that on CT scans of high data quality the traditional method is slightly better than the deep learning method as it is able to detect even smaller instances. But as

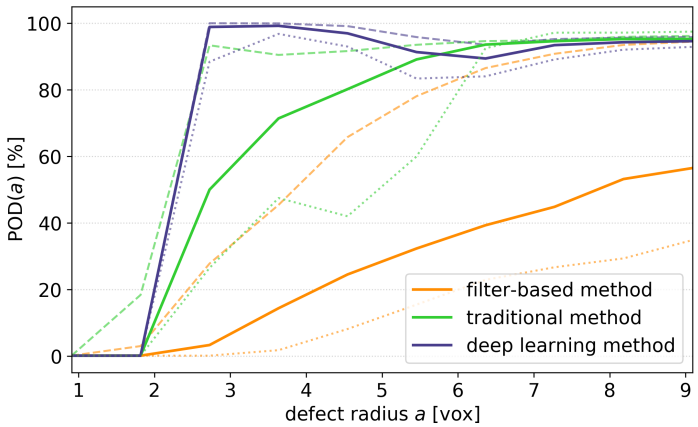


Figure 5.6.: What at first seems very confusing requires a brief explanation: We plot the POD which our reference-free defect detection algorithms achieve on the test specimen at different quality-levels. The solid line shows the POD for an average scan with a CNR of 9.8. We further plot the POD for a scan of better data quality (with a CNR of 14.8, dashed line) and a scan of worse data quality (with a CNR of 4.8, dotted line). We see that the deep learning method yields robust results for a wide range of data qualities (the POD curves are close together), while the traditional method starts to loose smaller defects with a decrease in the data quality. The filter-based method struggles heavily as soon as the data quality slightly changes. The drop we observe at larger defect sizes in the curve of the deep learning method is due to a slight under-segmentation of larger defects. These results are computed using the CT scans of our test specimen.

soon as the data quality recedes, the POD curves flatten, shifting a secure detection towards larger instances. When looking at the filter-based method, this effect becomes even more apparent. A wide gap opens up between the best and the worst data quality we evaluate the method on.

With the test specimen a more controlled evaluation is possible and we can determine a more reliable $a_{90/95}$ value. The deep learning method achieves an $a_{90/95}$ value of 5.6 voxels for the CT scan with a CNR of 9.8 and the traditional method an $a_{90/95}$ value of 9.3 voxels. However, the filter-based method still requires a better scan quality for a $a_{90/95}$ to be computable. For the CT scan with a CNR of 14.8 it achieves an $a_{90/95}$ value of 12.7 voxels which still is far behind the machine learning methods. One drawback of this special test specimen is that it does not consider all the artifacts which might occur in production. In particular, streaking artifacts arising from the partial volume effect are neglected completely and the scattering probably is different for the actual object.

5.3. Intersection over Union

The IoU (or Jaccard-index) is a voxel-based measure telling us more about the overall quality of our segmentation mask [205]. Yet, it does not provide any information about whether a loss in IoU arises from missing out one large instance, from missing out many tiny instances with the same combined volume, or from introducing a lot of false positive responses, i. e. the IoU is biased towards large instances [17]. Therefore, we also have a brief look at the instance-based intersection over union, which further weights the contribution of each voxel by the size of the instance it belongs to.

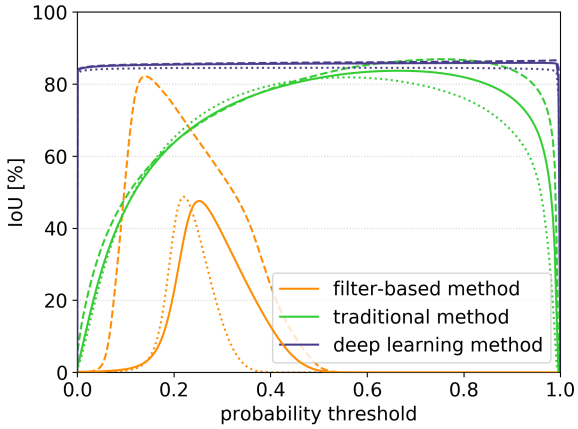


Figure 5.7.: The IoU as a function of the threshold which is used to create the binary segmentation from the probability map. The solid line shows the average IoU, the dashed line the IoU achieved on the CT scan with the best data quality, and the dotted line the IoU achieved on the CT scan with the worst data quality.

5.3.1. Computation of the Intersection over Union

As its name suggests, the computation of the IoU is less tedious than computing the POD: It is the intersection of the two binary masks which we like to compare, i. e. the ground truth and the prediction, divided by their union (see Equation (5.2)) [205].

$$\text{IoU} = \frac{\text{prediction} \cap \text{ground truth}}{\text{prediction} \cup \text{ground truth}} = \frac{TP}{TP + FP + FN} \tag{5.2}$$

Our predictions are non-binary probability maps. Therefore, we can plot the IoU as function of all possible thresholds to binarize the probability map (see Figure 5.7). This plot not only provides us with the best threshold to create the binary segmentation mask, it further gives us a notion of how accurate and how crisp the probability map is. If the plot shows us a wide plateau, the segmentation algorithm yields a very crisp segmentation mask. If the IoU of this plateau, in addition, is at a high level, we, furthermore, have a very precise segmentation mask, as it is the case for the deep learning method. Here, we almost can choose an arbitrary threshold to binarize the segmentation mask. In contrast, if we only have a narrow peak in the plot, it indicates a more unspecific and fuzzy segmentation mask, as it is the case for the filter-based method. Here, we have to be very careful in choosing the right threshold. Additionally, we see that the filter-based method has the largest loss in prediction performance when turning from CT scans with good data quality to ones with lower data quality. The deep learning method handles this best throughout our validation set with all the results being very close together, almost independent of the data quality. For the traditional method we see a gap opening up between the two curves which indicates that the confidence of the method declines as the data quality decreases and the segmentation of the defects becomes imprecise. We observe a similar behavior on our real CT scans. In Table 5.1 we compare the IoU our methods achieve on our realistically simulated validation set to the IoU they achieve on our two labeled real CT scans. We see that the relative drop in prediction performance between simulated and real data is about the same for all our methods—learning based or not. We have two explanations for this: Either the synthetic data misses crucial aspects of the real world, or the labels of the real data are not precise enough

and limit the prediction performance. Nevertheless, it can be seen that the learning based methods, which are trained solely on our simulated training set, outperform the filter-based approach. We, therefore, argue that the precise labels of our synthetic data pay off.

method	simulation	real data
filter-based	48.5 % (82.1 %/47.2 %)	40.7 %, 29.9 %
traditional	82.6 % (**86.9** %/81.0 %)	56.6 %, 43.5 %
deep learning	**85.9** % (86.6 %/**85.3** %)	**60.7** %, **47.3** %

Table 5.1.: The table shows the maximum IoU which our methods achieve. The first column shows the results of our simulated data: average (best/worst). The second column shows the results of our real CT scans: aluminum part, pivot cap.

To give a brief insight into how the defect segmentation looks like, we show eight sample patches with the results of our three reference-free defect detection algorithms in Figure 5.8. Four patches are taken from the CT scan of the aluminum part and four are taken from the CT scan of the pivot cap. These patches illustrate the issues which separate our methods from being perfect classifiers. For the filter-based method we see that it is particularly challenged with image noise. In the patches of the aluminum part the defects are slightly under-segmented; however, by lowering the threshold we would introduce too many false positives in the noisy areas of the CT scan. To achieve an optimal IoU we, therefore, have to live with under-segmentation and missing the tiny defects. In contrast, in the CT scan of the pivot cap we only obtain a good segmentation of the defects by accepting a lot of false positive responses. The data quality is too bad to find a good globally optimal threshold for the results of the filter-based method. The machine learning methods—the deep learning method in particular—are good in dealing with the noisiness of the data. However, the traditional method tends to over-segment defects in noisy environments. Nevertheless, the traditional method also yields indications for tiny defect instances, while the deep learning method seems to completely ignore defects smaller than a minimum size of 2 voxels.

With evaluating the POD and the IoU of our reference-free defect detection methods on simulated and real data, we demonstrate both the eligibility of our realistic simulations as training data for machine learning models as well as the advantages of our machine learning methods. Especially for CT scans of low data quality, the deep learning method yields very promising results.

5.3.2. Artifact Space Evaluation

Our synthetic validation set comprises two virtual cast parts, for which we have 27 realistically simulated CT scans with three gradations in each dimension of our artifact space (noise, beam hardening, and ring artifacts). We can use those CT scans to further examine the influence individual artifacts have on the automated detection of defects and their precise segmentation. We separate the prediction results along one axis of the artifact space and compute the mean IoU as well as its standard deviation over the remaining axes. In Figure 5.9 we show how the prediction results degrade when increasing noise, beam hardening, and ring artifacts individually.

From Figure 5.9a, we see that noise has the most considerable influence on the prediction performance. The mean IoU drops significantly as the noise-level increases. Moreover, the standard de-

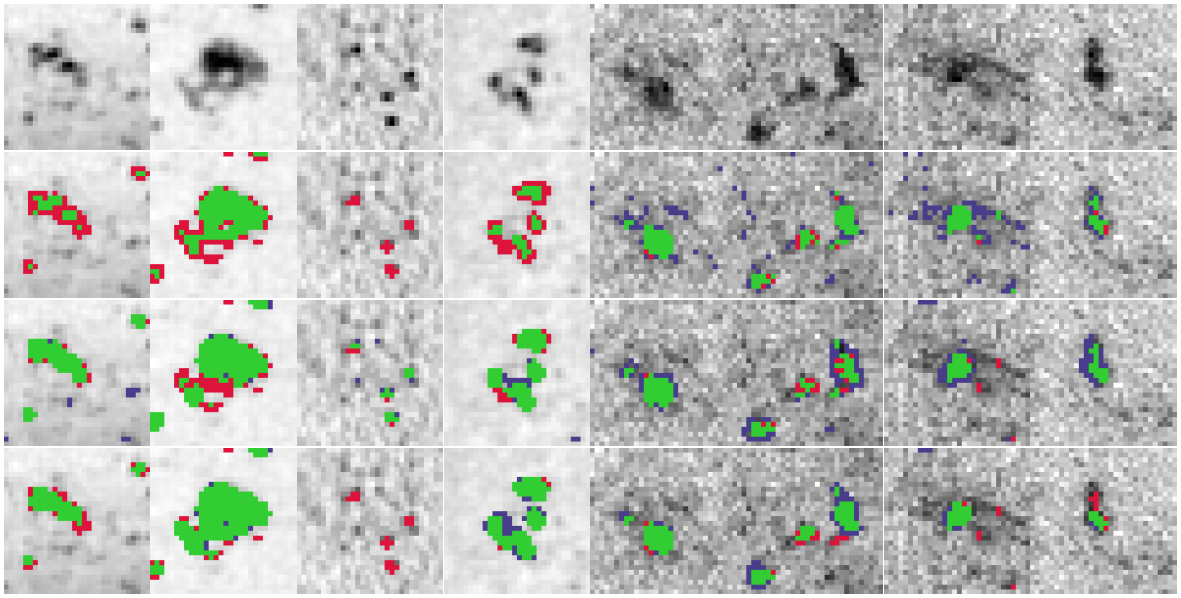
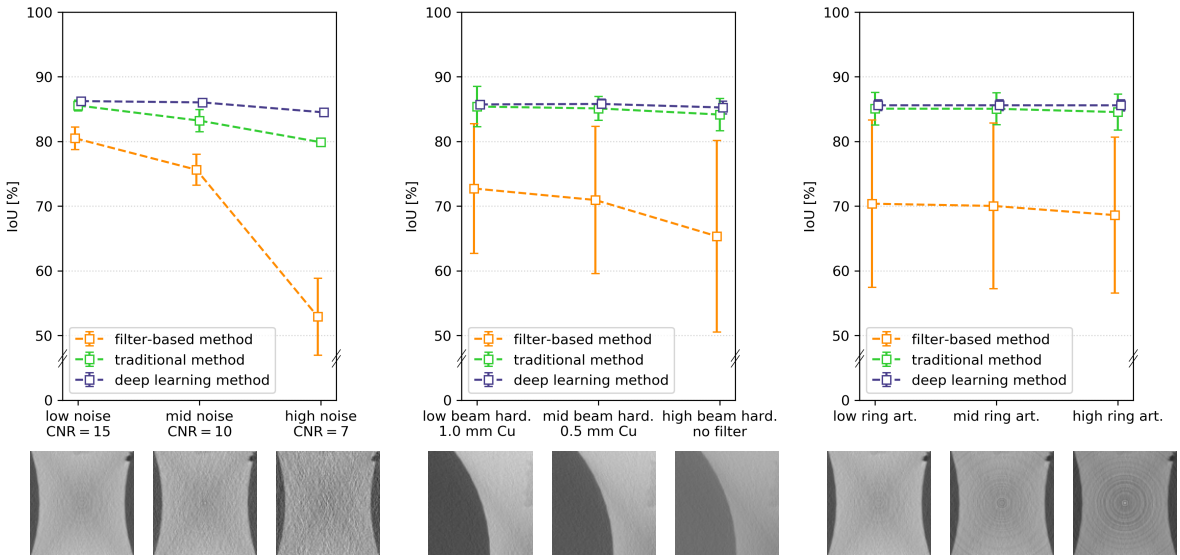


Figure 5.8.: Prediction performance on real CT data. From top to bottom: Original gray-values, results of the filter-based method, the traditional method, and the deep learning method. In green we show the true positive predictions, in red the false negatives, and in blue the false positives. We see that the filter-based method especially struggles with the noisiness of the data. Here, it is particularly difficult to weigh up precision and recall. We either miss out tiny defects (patches on the left) or introduce false positive responses (patches on the right). The traditional method is able to contain the number of false positive responses but still has some problems with noisy data. The deep learning method, in contrast, only produces very few false positives most of which result from a too generous labeling. Yet, it seems to have problems finding the smallest defects in this data set.

viation notably increases when moving from low noise-levels to higher ones, which means that with a high noise-level other artifacts become more impeding, too. The filter-based method particularly suffers from image noise, while the machine learning methods, most notably the deep learning method, are less sensitive. We observe the same effect for cupping artifacts which arise from beam hardening, albeit less significant. The drop in terms of the mean IoU is smaller when increasing the beam hardening, but we have an overall higher standard deviation (see Figure 5.9b). This is because the results now include all noise levels. We also observe a slight increase in the standard deviation when increasing the effect of beam hardening, which suggests that the impact of other artifacts increases when the cupping artifacts become more severe. Again, the filter-based method suffers most from this type of image artifact. Ring artifacts seem to have the least significant influence on the prediction performance. The mean IoU barely drops, even for predictions of the filter-based method and the standard deviation stays at about a constant level. The high standard deviation is explained by the influence of the two other types of image artifacts.

Up to now we keep the strength of the individual artifacts within the scope of a meaningful CT scan as it can appear in the everyday life in the quality laboratory. With our fully automated simulation pipeline it is conceivable to further investigate the impact of individual artifacts on the detection algorithms beyond the current limits.



(a) noise (b) cupping artifacts (c) ring artifacts

Figure 5.9.: Insights from the artifact space: As we simulate the different artifact types at different strengths for the same virtual cast parts, we can further examine the impact an individual artifact-type has on the prediction performance. For this, we project all results onto a single dimension of our artifact space and plot mean and standard deviation of the IoU. This reveals the influence of the different artifact types on the outcome. We see that noise clearly has the most significant influence on the detection of defects. Whereby, the deep learning method seems to be very robust against noisy data. Cupping artifacts impede the precise segmentation, too. Machine learning methods, however, clearly benefit from the statistics they draw from similar data of the training set. Ring artifacts have the least influence, probably because severe ring artifacts only rarely occur around the rotation axis of a scan. The small patches beneath the plot visualize the strength of the respective artifact type. Note that we cut the y-axis so that only the relevant part from 50 % and above is displayed.

5.3.3. Instance-based Intersection over Union

Augmenting the IoU with the size information of the individual instances gives us the instance-based intersection over union (iIoU) [17]. This measure is more complex to calculate and harder to interpret as we have to take the sizes of the individual instance into account. It removes the bias towards larger instances from the IoU by weighting the contribution of each instance i in relation to the size of the average instance so that smaller instances gain higher weights and larger instances lower weights [17]. Compare Equation (5.3), where s_i denotes the size in voxel of instance i and N is the total number of instances.

$$\text{iIoU} = \frac{\sum_i \omega_i TP_i}{\sum_i \omega_i TP_i + FP + \sum_i \omega_i FN_i}, \quad \text{with } \omega_i = \frac{\frac{1}{N}\sum_n s_n}{s_i} \tag{5.3}$$

In Figure 5.10 we plot the iIoU as a function of the probability threshold as we did for the IoU. The iIoU draws a different image. In terms of the iIoU the traditional method seems to outperform the deep learning method, as it also detects instances which are smaller than 2 voxels in diameter. This, however, only is possible by accepting more false positive responses. Due to the crisp segmentation the deep learning method has not that much leeway. We see this in the drop of precision when we

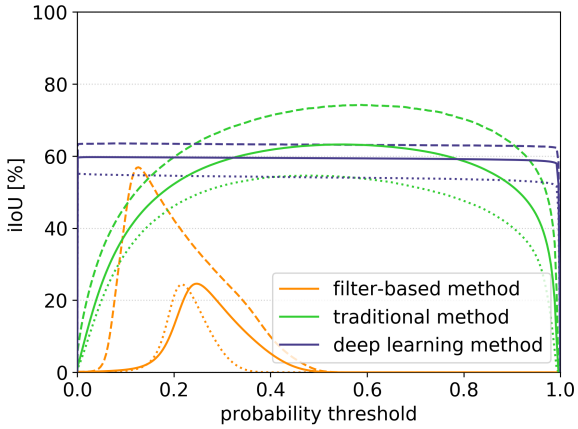


Figure 5.10.: The iIoU as a function of the threshold which is used to create a binary segmentation from the probability map. The solid line shows the average iIoU, the dashed line the iIoU achieved on the CT scan with the best data quality, and the dotted line the iIoU achieved on the CT scan with the worst data quality.

use the threshold that maximizes the iIoU. For the filter-based method, the traditional method and the deep learning method we get a precision of 85.9 %, 88.1 %, and 89.0 %, respectively. Interestingly, the differences between the CT scan with the best data quality and the one with the worst data quality is increased for all methods. This probably is because for CT scans with worse data quality detecting smaller defects is more challenging. Table 5.2 further shows the results for the real CT scans.

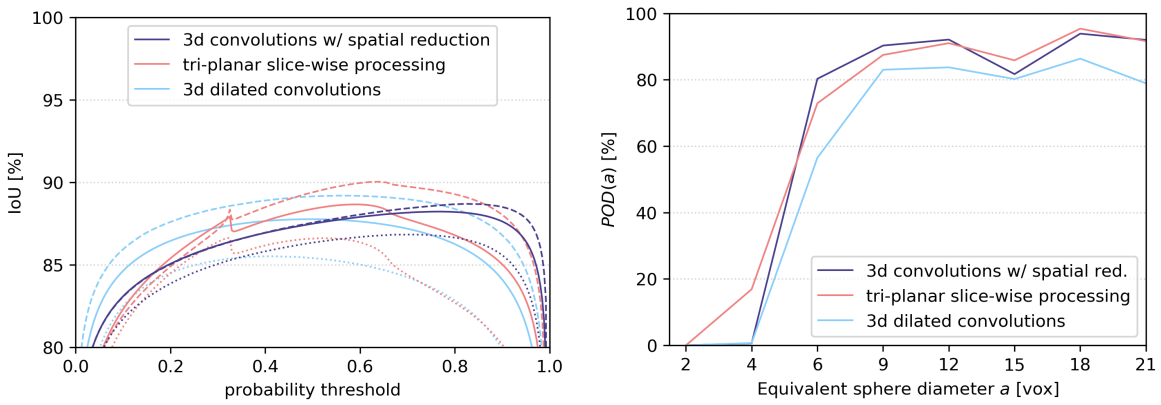
method	simulation	real data
filter-based	24.6 % (56.9 %/24.4 %)	38.5 %, 23.8 %
traditional	**63.2** % (**74.2** %/54.6 %)	**53.5** %, **30.8** %
deep learning	59.7 % (63.5 %/**55.1** %)	42.7 %, 26.1 %

Table 5.2.: The table shows the maximum iIoU which our methods achieve. The first column shows the results of our simulated data: average (best/worst). The second column shows the results of our real CT scans: aluminum part, pivot cap.

While the deep learning method loses ground for CT scans of good data quality, it still can make good use of its strength for CT scans of low data quality. The deep learning method yields very promising results especially for CT scans of low data quality (see Section 6.1).

5.3.4. Comparing Different Model Architectures

As mentioned in Section 4.3.1 we evaluated different model architectures. In Figure 5.11 we show a quantitative analysis of these model architectures in terms of the IoU and the POD. We evaluate (i) the architecture which comprises three-dimensional convolutions and context aggregation through spatial reduction; (ii) the slice-wise processing of the input along orthogonal slices using the U-Net architecture and an element-wise maximum to combine the results of the different directions; and (iii) the architecture that uses three-dimensional dilated convolutions for context aggregation.



(a) intersection over union

(b) probability of detection

Figure 5.11.: The quantitative results of the model architectures which we examined in Section 4.3.1 are comparable. The model which uses dilated convolutions to aggregate context information is slightly more vulnerable to changes in data quality and needs larger defect sizes for a proper detection. The model which uses spatial reduction to aggregate context information is least vulnerable to changes in data quality. The element-wise merging of the individual results of the tri-planar slice-wise processing approach using a maximum operation yields two spikes in the IoU. The slice-wise processing also detects some of the smaller defects. Note that these results are from a different iteration of our validation set and, therefore, slightly differ from the results presented before. The dashed lines corresponds to the results for the best data quality, the dotted lines to the worst data quality and the solid lines represent the average over all data qualities.

On average the different model architectures yield about the same IoU (88.2 %, 88.6 %, and 87.8 %, respectively). The slice-wise processing even manages to slightly outperform all other models on CT scans with a high data quality. However, the model architecture which aggregates the necessary context information through spatial reduction slightly outperforms the other models on CT scans of bad data quality. The slice-wise processing and the model which uses dilated convolutions for context aggregation seem to be more vulnerable to higher artifact-levels. The difference between their results on the CT scan with the best data quality and the one with the worst data quality is significantly larger (see Figure 5.11a). In addition, for the slice-wise method we observe two spikes in the plot which probably arise due to the element-wise merging of the three individual results using a maximum operation. In terms of the POD (Figure 5.11b) the slice-wise processing manages to detect at least some of the smaller defects (which have an ESD of 4.0 voxels or less). The POD of the dilated convolution model constantly is slightly below the POD of the other models. While the model which aggregates its context information via spatial reduction does not detect any defects with an ESD lower than 4.0 voxels, it achieves a higher POD for larger defects than the other models.

5.4. Precision Recall Curve

A crucial step to obtain a good segmentation from the probability output of the defect detection method is to choose the right threshold. Here, we need to weigh the sensitivity of a classification or segmentation system and the amount of false positives it introduces. The ROC curve offers a tool that allows us to tune the hyper-parameters of a system by plotting the true positive rate and the false negative rate while varying a hyper-parameter, e. g. the threshold to binarize the probability

output [92, 213]. Additionally, it provides a more general overview of the overall performance of the respective method. However, the ROC takes into account the true negative predictions, which does not allow for a meaningful comparison of our defect detection algorithms due to our highly imbalanced data set. All the curves go through the upper left corner, far from the random classifier which is represented as a diagonal from (0.0, 0.0) to (1.0, 1.0). This suggests the methods to have all a very high prediction performance, even though there are vast differences between them (see Figure 5.12a). Therefore, we compute the equivalent of the ROC for imbalanced data, the PR curve. It shows the relation of "how many of the voxels which are labeled as 'defective' actually are 'defective' ", i. e. the *precision* (see Equation (5.4)), and "how many of the total 'defective' voxels are actually labeled as 'defective' ", i. e. the *recall* (see Equation (5.5)), while varying the threshold which is used to binarize the prediction [210, 211] (see Figure 5.12b). The baseline in a PR curve, i. e. a random classifier, is determined by the proportion the positive class represents in the data set, while the curve of a perfect classifier would go through the upper right corner, i. e. finding all voxels of the target class and not yielding any false positive responses. In our case the baseline is very close to zero, at 0.02 % and, therefore, not explicitly included in the plot.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.4}$$

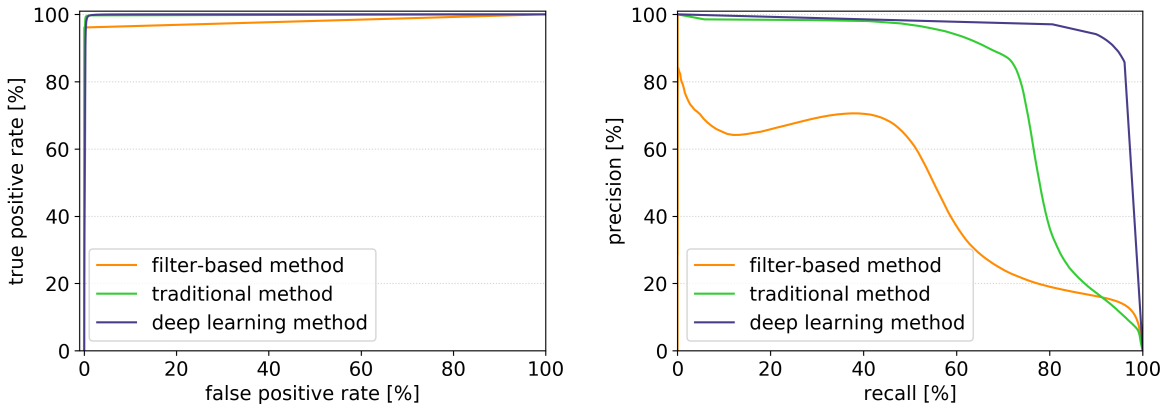
$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.5}$$

From the IoU plot in Figure 5.7 we already know that the deep learning method only predicts very confident results. Therefore, there is not much change when varying the probability threshold. This also reflects in the PR curve which we show in Figure 5.12b. With the deep learning method we have a very strong and confident classifier. For the traditional method we see that as soon as we increase the recall to find more and more defects we rapidly start losing precision, i. e. we get more false positive responses from image artifacts. The filter-based method never reaches a high precision, because at low recall the segmentation of the larger defects is imprecise as they are under-segmented, and when increasing the recall we certainly obtain a better segmentation but also introduce much more false positive responses.

A way to reduce the PR curve to a single performance measure is to compute the area under curve (AUC) [210]. A better classifier reaches farther out to the upper right corner and, therefore, covers more area of the plot. For the filter-based method, the traditional method, and the deep learning method, we achieve an AUC of 0.47, 0.78, and 0.96, respectively. If we would compute the AUC with only few support points, it would be important to consider that the interpolation between two points of the PR curve is non-linear [212].

5.5. Comparisons with Destructive Methods

From Chapter 3 we know that labeling real CT scans is an almost impossible task. Thus, it is correspondingly hard to carry out a reliable quantitative evaluation on real CT scans. Trying to obtain a better ground truth for validation, we call in another imaging domain: We switch from NDT methods to destructive ones; we cut the aluminum part in half (after doing the CT scan) and label the defects in the *cut image*. The cut image has a high spatial resolution as well as a very high contrast resolution (compare Figure 5.13a). This makes the labeling of defects much easier than in a CT scan—especially since there is only one layer to label. However, we still cannot use this method to



(a) receiver operating characteristics curve

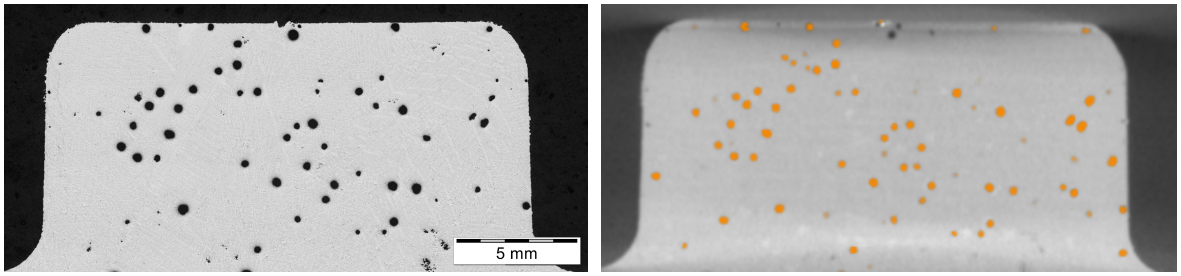
(b) precision recall curve

Figure 5.12.: To determine an optimal operating point of a classification (or segmentation) system, the PR curve provides a tool for imbalanced data sets (b), where the ROC curve would not be meaningful (a). The better the classifier the farther the PR curve reaches towards the upper right corner. A random classifier would form a straight line at the precision which corresponds to the portion of the target class (0.02 % in our case). The deep learning method is the most confident method. The traditional method rapidly looses precision when reaching a certain level of recall, indicating it introduces more and more false positive responses. The filter-based method struggles with under-segmentation for low recall and suffers from high false positive rates at higher recall values.

create enough training for a proper training. For a conclusive evaluation and a comparison to long-established destructive methods a single labeled slice is sufficient. In this section we compare the results of a cut image to the results which our deep learning method achieves on the corresponding CT scan.

The Austrian institute for foundry (ÖGI) has done these measurements. As a test specimen, they used a stepped pyramid in which they artificially placed defects by foaming the melt [217]. Then, after performing a CT scan and cutting open the pyramid, they made a two-dimensional image of the cut surface (see Figure 5.13a). In order for us to further evaluate our method, the ÖGI have thankfully shared this data with us. Still, we find it quite hard to do a quantitatively meaningful comparison. The hardest part is to correctly align the cut image with the CT scan. The cut image has a much higher spatial resolution. We, therefore, see much smaller defect instances which are not visible in the CT scan. Moreover, the voxel grid is not aligned with the cut. Thus, we have to interpolate between voxels in order to obtain a slice which corresponds to the cut image. Finally, due to the underlying physics of the imaging process, the CT scan is more blurry, which means defects from adjacent slices shine through on our target slice and so do our predictions. These defects, however, are not cut open in the cut image and, in consequence, are not visible. We align the cut image with the CT scan as best as possible, which enables us to see a clear correlation and allows at least for a qualitative comparison (see Figure 5.13b). A quantitative comparison, for instance, using the IoU measure is not advisable as the slightest misalignment of the cut image and CT scan has a tremendous effect on the IoU.

Hence, we move to a comprehensive, derived measures, for a quantitative comparison. We measure the overall porosity of the cut image and of the corresponding region in the CT scan which is close by. In the analysis of the ÖGI, the cut image shown in Figure 5.13a has a porosity of 2.2 %. To be more



(a) a cut image of a step pyramid [217]

(b) about the same slice in the corresponding CT scan

Figure 5.13.: Another way to evaluate a novel defect detection and segmentation algorithm would be the comparison to established methods like the destructive method of using cut images (a). However, this comparison is challenging in multiple ways. (i) First of all the cut image usually has a much better spatial resolution than the CT scan which makes it possible to even detect smallest defects. (ii) It is hard to correctly align the slice with the CT scan. (iii) Due to the physics of the imaging process the CT scan is more blurry than the cut image. Therefore, defects from adjacent slices which are not visible in the cut image shine through. Nevertheless, we can see correlation between the cut image (a) and the corresponding slice in the CT scan (b). While it allows a brief qualitative comparison, using it for a meaningful quantitative evaluation, however, would be questionable. The orange overlay in (b) represents the prediction output of our deep learning method.

precise, 2.2 % of the material part are pores. For about the same region in the CT scan we observe a total porosity of 2.4 % which matches the porosity determined by means of the cut image.

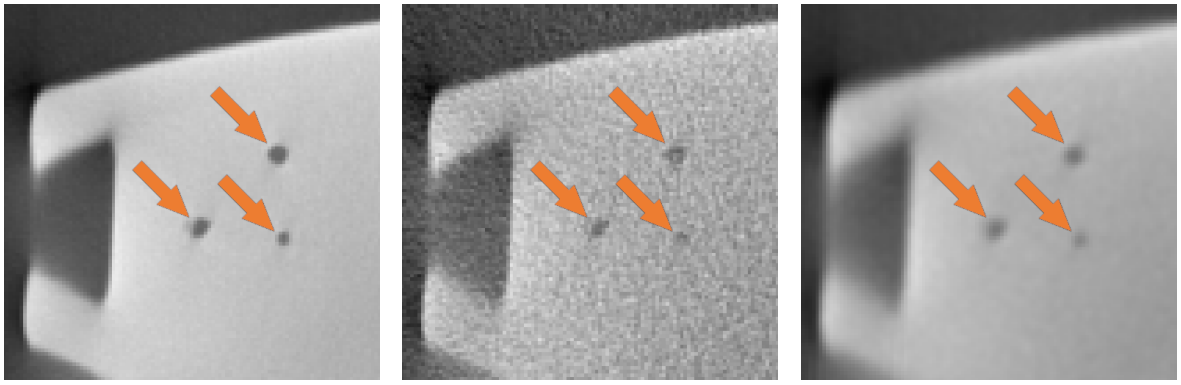
The first evaluations of our reference-free defect detection (and segmentation) methods are very promising: By showing that all three methods—learning-based or not—show a drop in prediction performance when switching from our realistically simulated data to real CT scans, we argue that our simulated data set is sufficient to train a machine learning model for the detection of defects in CT scans of cast aluminum parts. Moreover, we observe that the deep learning method yields the most robust results with the least false positive responses. Yet, for CT scans with a sufficient data quality the traditional machine learning method slightly outperforms the deep learning method by finding smaller instances. Nevertheless, for CT scans with low data quality the deep learning method yields the best segmentation. So far we only considered the precise segmentation of the defects. In Chapter 6 we will put deep learning through its paces and in Section 6.3, in particular, we will extend it to distinguish between different types of defects. We further evaluate how well the deep learning method performs in in-line scenarios in which we have to deal with especially challenging data and we examine what happens when altering the modality of the input data.

6. Discussing Deep Learning

In the previous chapter we introduced the tools to evaluate our reference-free defect detection methods and have shown that (a) our realistic simulations serve well as training data, in particular because of their precise labels, and that (b) the deep learning method is able to outperform other methods, especially on CT scans of poor image quality. However, we not only need to beat another high score [26], we like to develop a robust and reliable deep learning product which enables quality specialists to solve the inspection tasks of their daily routine more easily. With this premise in mind, in this chapter, we further explore the possibilities and limitations of this approach. As we observe the deep learning method to be particularly promising in CT scans of low data quality, we evaluate the performance of our method in in-line scenarios, in Section 6.1. Here, the inspection has to keep up with tight production schedules, which means there is only little time to acquire the data necessary for a reliable decision. The CT scans look accordingly: noisy and artifact-prone [42, 43]. With our deep learning method not only being very precise but also quite fast in inference, we should be capable of obtaining robust results and staying within the timing constraint. In Section 6.2 we examine how we can adapt our model which we designed to segment defects in CT scans of cast aluminum parts to different applications. Here, we encounter particularly challenging CT scans as the materials are hard to penetrate or the target is surrounded by a material which is hard to penetrate and consequently only shows a very limited contrast resolution. Inspection tasks often not only require to detect and to precisely segment defect instances but also to classify them. For example, cavities are considered more harmful regarding the stability of a part than gas pores [1]. We tackle the subsequent classification of defects in Section 6.3. Finally, a major concern of some users is the question what happens if they deviate from the intended use case. Will the model confidently predict false positive results or will it confidently ignore true positive incidents? And how can this be avoided? Hence, we include the model uncertainty in our analyses in Section 6.4, to provide the user with further information about the trustworthiness of the results. While we got along with only one training set, one validation set, and two real CT scans in the last chapters, we introduce plenty of new data sets for training and for validation as well as further real CT scans throughout this chapter. In Appendix A, we, therefore, provide an overview of all the data sets mentioned in this chapter and an overview of all the models trained in Section 6.1 and Section 6.2 in Appendix B.

6.1. Eligibility for In-line Scenarios

When moving from a random sample inspection to a full in-line inspection of the entire production, the time constraints become much tighter and, hence, the artifact-level of the CT scans increases. Often, we only have up to two minutes for a CT scan of a part which has the size of a typical car cylinder head [44]. Unfortunately, the acquisition of a CT scan which allows for an uncomplicated inspection is time consuming: it is necessary to take a large amount of projections to reduce sampling artifacts and to use long exposure times per projection to mitigate image artifacts, noise above all



(a) A CT scan from the quality laboratory using an exposure time of about 1400 s at 225 kV and 2 mA.

(b) The same part scanned under in-line conditions using an exposure time of only 120 s.

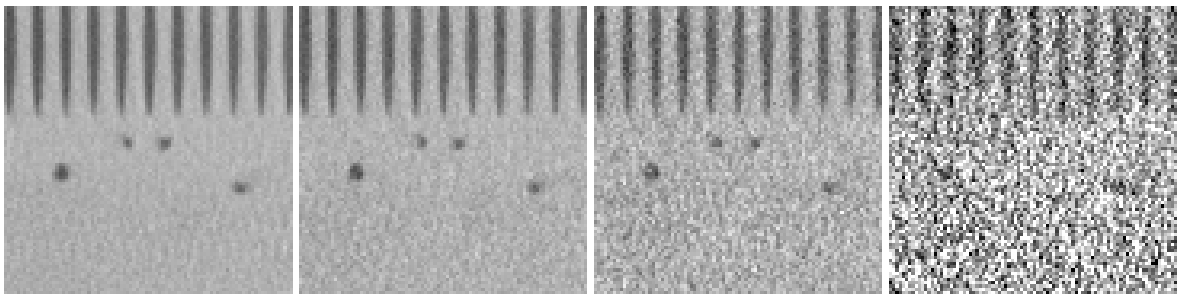
(c) The CT scan after trading spatial resolution for contrast; scanned in 120 s but at 320 kV and 3 mA.

Figure 6.1.: CT scans of varying image quality. The orange arrows indicate the positions of the same features in each scan. To increase the contrast in fast CT scans, we can go to higher energies so that more photons arrive at the detector. This, however, leads to a lower spatial resolution, mostly caused by large focal spot sizes.

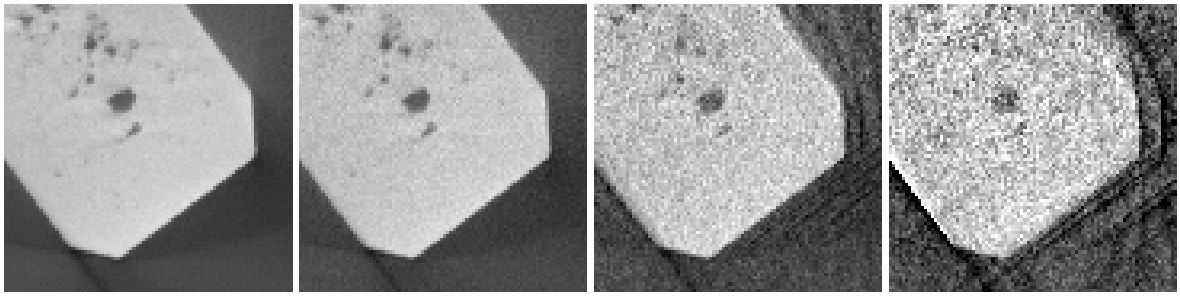
[35, 43, 215]. The low data quality of the CT scans affects the semantic segmentation of the defects. To provide an example, Figure 6.1a shows a slice of a CT scan of an aluminum part as it could occur in the quality laboratory. The scan is taken with 225 kV at 2 mA using an exposure time of 1 s per projection. When examining a random sample in the quality laboratory, we usually have the time to take care of both, the spatial resolution and the contrast resolution. We mitigate the artifacts arising from beam hardening by using a pre-filter of 3 mm of copper. In total we acquire 1440 projections. The slice in Figure 6.1b shows the same part. This time, the CT scan is done within the time limit of two minutes. While we maintain the spatial resolution, the *contrast resolution* significantly suffers from the short scan time and we have to deal with a high noise-level. Furthermore, we reduced the pre-filter to 0.5 mm of copper to increase the amount of photons arriving at the detector. The defects, especially the smaller instances, are almost lost in the image noise. By trading *spatial resolution* for contrast resolution, we keep the noise-level low. This requires to increase the power (or rather the voltage) of the source, which on the other hand results in an increased focal spot size. When keeping the time constraint fixed and going for higher contrast resolution, the spatial resolution will suffer inevitably [42]. We increase the tube voltage to 320 kV. The CT scan becomes less noisy but appears to be more blurry (see Figure 6.1c). While the defects become more visible to human inspectors, their representation in the image domain of the CT scan changes: The transitions from material to background become wider, which affects the machine learning methods in particular because they are used to crispier transitions. Therefore, we first set up an experiment to see how well our reference-free defect detection methods handle CT scans of decreasing data quality and, then, test the deep learning method in a real in-line scenario.

6.1.1. Scan Time Reduction

Our machine learning methods, the deep learning methods in particular, yield promising results on CT scans of low data quality and, therefore, should be able to cope with the fast CT scans. To evaluate by how much we can decrease the scan time before we significantly differ from a precise segmenta-



(a) *Simulated data:* When reducing the total scan time by gradually decreasing the exposure time per projection, we observe an increase in noise. From left to right we use a total scan time of 720 s, 360 s, 180 s, and 7.2 s. Each CT scan is reconstructed from 720 projections.

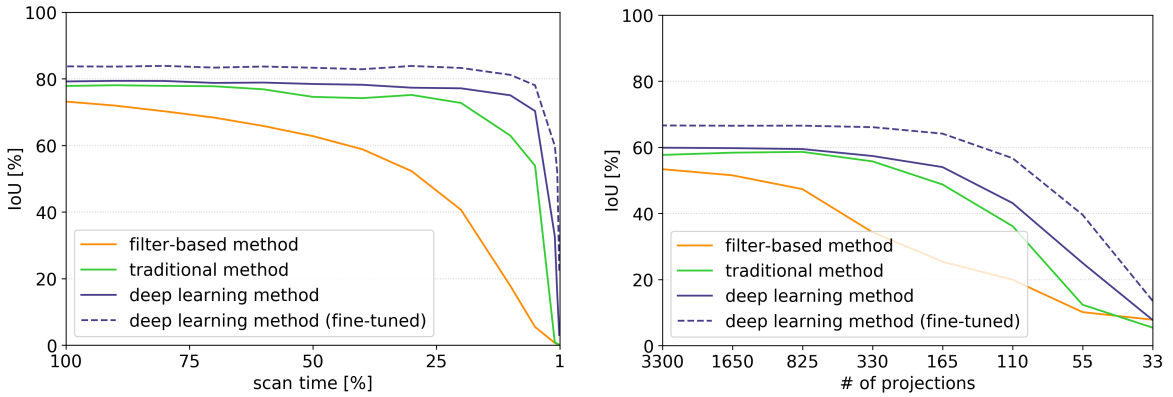


(b) *Real data:* When reducing the total scan time by reducing the number of projections, we do not only observe an increase in noise but also the emergence of aliasing artifacts due to under-sampling. From left to right we use 3300, 825, 165, and only 33 projections to reconstruct the CT scan. Each projection has an exposure time of 0.4 s.

Figure 6.2.: To reduce the total scan time, we can either reduce the exposure time per projection (a) or we can reduce the number of projections (b). Both decrease the total amount of information which is available in each voxel of the reconstructed CT scan. Reducing the number projection, in exchange, further induces aliasing artifacts to the CT scan.

tion, we simulate 14 CT scans while decreasing the exposure per projection (see Figure 6.2a). Thus, all CT scans have the same number of projections, i. e. 720. However, the overall simulated scan time is reduced from 720 s to 7.2 s. For this data set we use a CT setup from our training data with low beam hardening and medium ring artifacts, i. e. 225 kV at 1 mA with a pre-filter of 1 mm of copper. Then, we further vary the exposure time per scan. This directly reflects in the amount of noise we observe in the reconstructed CT scans. While even small defect instances are clearly visible for long exposure times, we can only guess if there is a defect in the fastest CT scan. In Figure 6.3a we plot the maximum IoU as a function of the relative scan time. We see that for the filter-based method the segmentation quality (in terms of the IoU) quickly declines as soon as we reduce the scan time. The machine learning methods are much more robust and are able to maintain a high IoU even for fast CT scans. At about a forth of the original scan time, the traditional method still achieves about the same segmentation quality as the filter-based method does for the full scan time. With the deep learning method we can even reduce the scan time to about 10 % and still obtain similar results.

To verify these results on real CT data we turn to the reduction of projections. Creating a repeatable CT setup is quite a hard endeavor. Minor fluctuations in source and detector as well as imprecision in the geometry and rotation of the setup influence the CT scan and the alignment of the part. Moreover, we cannot arbitrarily decrease the exposure times. Therefore, we take another high-quality CT scan of the pivot cap with many projections, i. e. 3300. Each projection has an exposure time of 0.4 s. Then, we reconstruct the volume 8 times with a decreasing amount of projections: 3300 (which



(a) IoU as a function of the relative scan time (evaluated on the simulated data shown in Figure 6.2a)

(b) IoU as a function of the number of projections (evaluated on the real data shown in Figure 6.2b)

Figure 6.3.: We plot the maximum IoU as a function of the total scan time. The results confirm what the analysis of the artifact space indicated: The machine learning methods are better in dealing with high image noise. While the IoU of the filter-based method declines quickly when the data quality is decreased, the machine learning methods can maintain a satisfying segmentation of the defects. Especially the deep learning method permits for very fast CT scans. In the experiment on the real CT scan (b) the IoU already declines at longer scan times than in our simulated experiment. We assume this is due to the aliasing artifacts which induce another challenging aspect to the CT data.

corresponds to about 1300 s), 1650, 825, 330, 165, 110, 55, and only 33 (which corresponds to a total scan time of only 13 s). This similarly reduces the total information (number of photons) we have per voxel. However, as soon as the angle steps between the projections become larger than the voxel size, we start under-sampling the object and introduce aliasing artifacts which manifest in the CT scan as shadows of the actual part (see Figure 6.2b). In the outer regions of the field of view this has a more severe impact than close to the rotation axis. In Figure 6.3b we equivalently show the IoU as a function of the number of projections used for the reconstruction. We observe a similar behavior of our reference-free defect detection methods as we do for the simulated data in Figure 6.2b. The IoU of the results of the filter-based method quickly starts to decrease as the data quality declines, while the machine learning methods are able to maintain robust results even for faster CT scans. In this experiment the drop in terms of the IoU for the machine learning methods happens a little earlier than in the experiment with the simulated data. The IoU already starts to drop at a scan time of about 66 s (or 165 projections). Probably, this is due to the increasing severity of aliasing artifacts which add another challenging aspect to the CT scans.

6.1.2. Fine-tune for Fast Scans

Usually, the detection of defects is not the only objective in an in-line scenario. The CT scans are, for example, further used for measurement tasks. These require a sufficient contrast resolution, because it is important that a structure is resolvable before it can be measured [43]. Therefore, if we like to develop a reference-free defect detection method which is as well applicable to in-line scenarios, we need to deal with CT scans which are optimized for contrast resolution (see Figures 6.1c or 6.5a). This means that these CT scans are significantly blurrier due to larger focal spot sizes and, hence, clearly differ from our original training distribution, which tries to maintain crisp edges

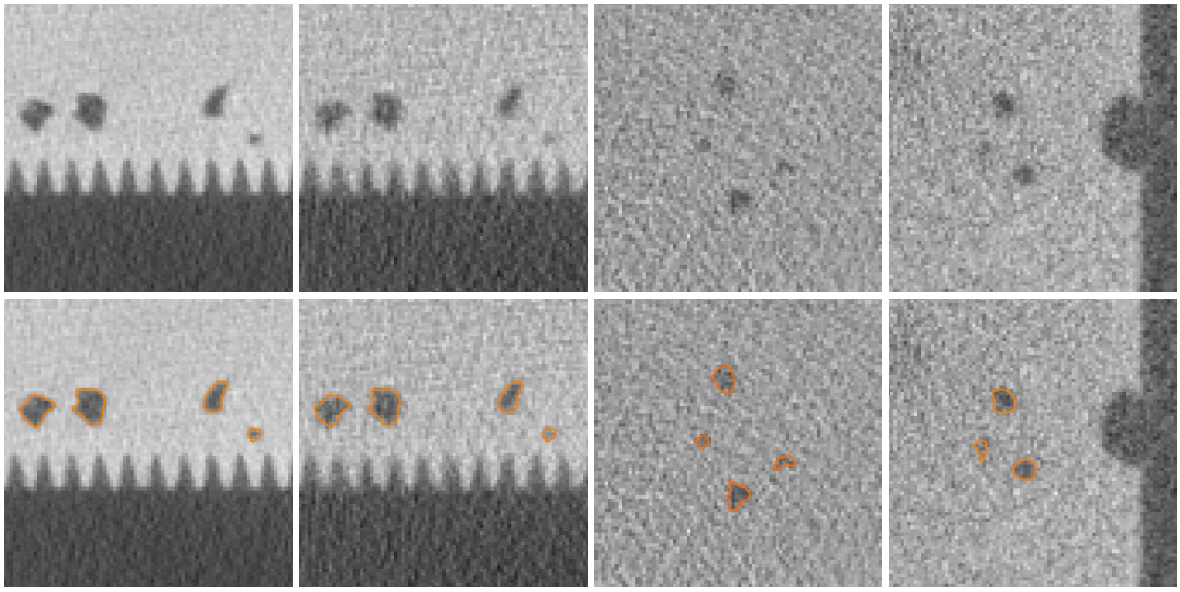
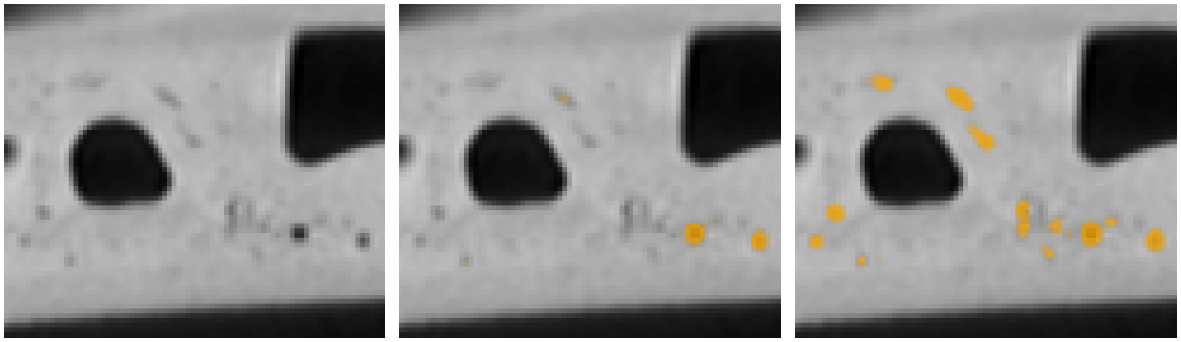


Figure 6.4.: Our initial training set was designed to resemble CT scans from the quality laboratory. CT scans from in-line scenarios, however, do not meet this data quality. To adapt our deep learning method, we augment our training set with additional CT scans which are made using large focal spot sizes and increased noise-levels. The additional training data is then used to fine-tune our model for the low data quality we face in in-line scenarios. The orange outlines show the precise boundary of the defects of the raw data in the top row.

using a relatively small focal spot size. The results of the deep learning method are correspondingly bad (see Figure 6.5b). To make up for the reduced spatial resolution we encounter in most in-line CT scans, we need to adapt our training distribution by augmenting our data set. We simulate 50 additional CT scans which resemble typical in-line CT scans. The additional data comprises CT scans with (i) an increased noise level, which we achieve by reducing the virtual scan time, (ii) increased cupping artifacts, which we achieve by reducing the tube voltage and the thickness of the pre-filter, and (iii) most importantly, with a reduced spatial resolution, which we achieve by increasing the size of the focal spot on purpose. Here, we benefit from our simulation pipeline, as we can intentionally tune the parameters in a way to obtain CT scans of low data quality even for small parts for which it would be hard to get a bad scan in real life. For the additional training set we operate the X-ray source at 120 kV without using a pre-filter. To increase the noise-level we vary the exposure time per projection between 0.06 s and 0.08 s. For the decreased spatial resolution we increase the focal spot size to a range between 320 µm and 400 µm. The increased focal spot size is justified by an increase in the current of the source to range between 0.6 mA (for the small focal spot sizes and long exposure times) and 1.0 mA (for the large focal spot sizes and short exposure times), respectively. In Figure 6.4 we show some example slices of the additional CT scans which augment our training data.

When preparing our deep learning method for its new field of application we fortunately do not need to train them from scratch. Studies suggest that CNNs tend to learn more common features like edge, color, or simple pattern detectors in their early layers, and only later, in the deeper layers, evolve more task specific features [115, 218]. We can take advantage of this behavior. We use the kernel weights of the model which we pre-trained on our original training set, i. e. the segmentation



(a) A slice taken from in-line CT data (b) The results of the original model (c) The results of the fine-tuned model

Figure 6.5.: From left to right: the raw data of a real in-line CT scan, the results of the original model, and the results of the model which was fine-tuned on simulated low data quality in-line CT scans. The orange overlay represents the prediction output of the respective models.

of defects in quality laboratory CT scans, and only adapt them to match with the in-line scenarios by using the additional training data, i. e. we *fine-tune the model* for in-line scenarios. This approach requires less training data. The model already has a notion of its task. Another benefit is that the adaptation helps the model to better generalize on its original task—as long as the tasks do not differ too much [115]. We fine-tune our original model with a reduced learning rate of 10^{-5} for another 25 000 iterations on the additional training data, adapting all upstream and refinement layers. Therefore, the fine-tuned model also outperforms all other models in terms of the IoU in our scan-time reduction experiments (see Figure 6.3) and it achieves significantly better results on our real in-line data (see Figure 6.5c).

The deep learning method detects and segments defects with a high precision even in fast CT scans of quite low data quality. The results are particularly good when fine-tuning the model for a specific scenario, i. e. for a specific part and a specific CT setup. Furthermore, using the appropriate hardware it is possible to keep up with the high cycle times of in-line CT setups. We are able to process a volume with 1000 × 1000 × 1000 voxels in about 500 s on a single GPU. The block-wise processing of the input further allows us to distribute the workload to multiple devices. The prediction time scales almost linearly with the number of devices—at least to a certain extent. With these promising prerequisites at hand, we evaluate what it takes to move from cast aluminum to different input domains in the following section.

6.2. Transferability to Other Applications

The inspection of cast aluminum parts is not the only application of CT in the world of NDT. In this section, we further examine how well our deep learning method performs on other comparable tasks. We mostly change the input domain, not the objective. Step by step we move away from cast aluminum in four small experiments: (i) Firstly, we change the material of the part under examination, evaluating the deep learning method on plastic injection molding and iron casting. (ii) Secondly, we examine multi-material parts because sometimes defects only emerge during the composition of individual parts. The inspection becomes especially challenging, if the individual parts widely

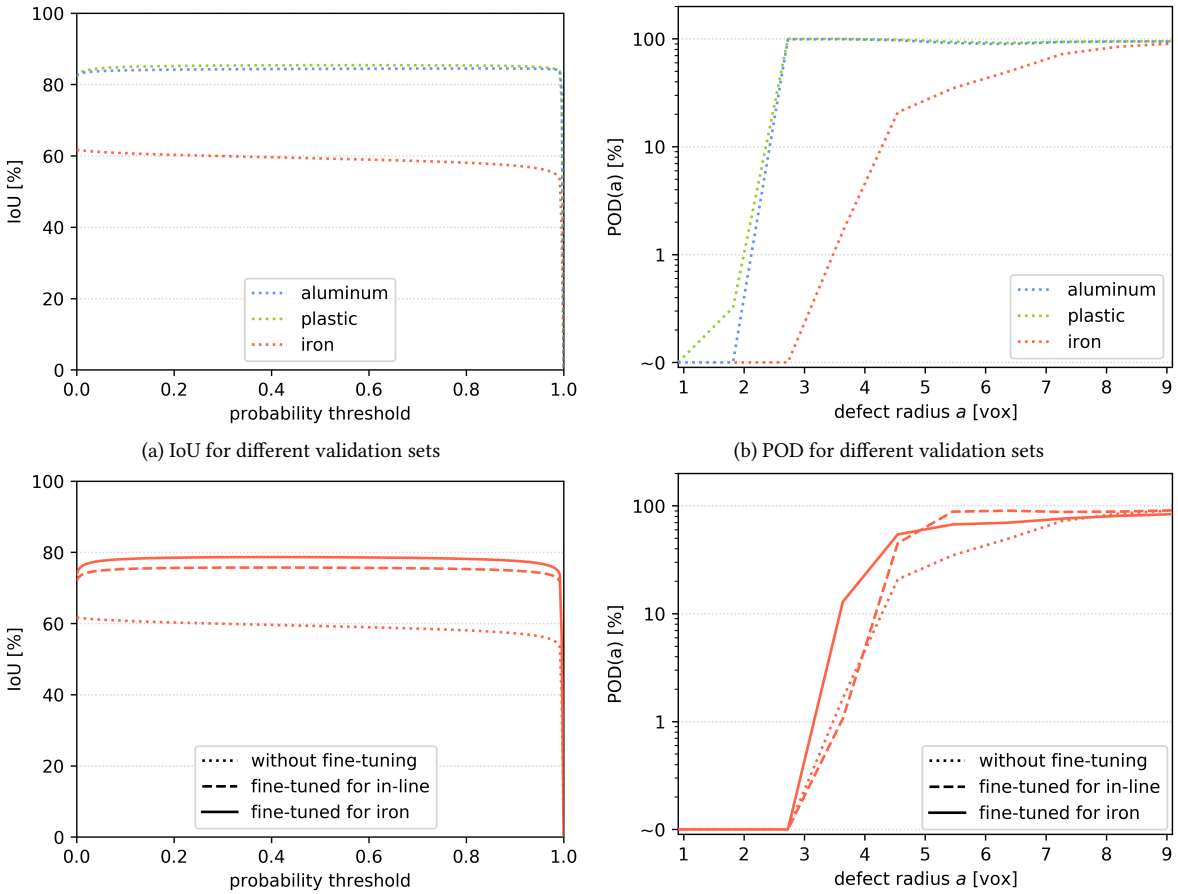
differ in their penetrability. (iii) Then, we change the manufacturing technique of the part under examination, evaluating an additively manufactured part. This comprises two challenges: the defects usually are smaller and their detection is more difficult because the parts are made of materials of high density which leads to a low data quality of the CT scans. (iv) Finally, we evaluate how we can modify the objective by examining the detection of thin, large cracks in nickel-based alloys, which, for example, emerge during fatigue testing.

6.2.1. Plastic Injection Molding and Iron Casting

Gas pores and shrinkage cavities not only occur in light metal castings but, for example, also in parts of injection molded plastic or cast iron. If we neglect that the actual defect shapes might slightly change due to the differences in the production process and the material properties, we find the most significant difference regarding the quality assessment is the change of the data quality of the CT scans. While in general plastics are easier to penetrate than aluminum, iron has a much higher density and is way harder to penetrate, which leads to an increased *artifact-level* in the CT scans. To evaluate the influence of these artifacts on the precision of the defect segmentation, we simulate the virtual cast parts of our validation set again. This time we use polyoxymethylene (POM) creating our validation set for injection molded plastics and the iron alloy EN-GJL-150 [219], which also includes portions of carbon and silicon, creating our validation set for cast iron. In Figure 6.6 we show the quantitative results of our deep learning method on the different validation sets of different materials and how fine-tuning can improve the results for the particularly artifact-afflicted iron cast data. To simulate the plastic CT scans, we use a virtual source with 120 kV at 0.1 mA without any pre-filter. With this setting, a focal spot size of 160 µm × 160 µm should be possible, resulting in crisp edges and a high spatial resolution (see Figure 6.7a). In contrast, to obtain a reasonably comparable data quality for the parts of cast iron we need to operate our virtual source with 600 kV at 0.6 mA and use a pre-filter of 2.5 mm copper. As a result, the focal spot size is increased to 320 µm × 320 µm, which then again results in a significantly lower spatial resolution (see Figure 6.7b).

As we neglect possible shape deviations which could occur in different materials, our simulated validation sets mostly differ in the data quality of the CT scans. Hence, the loss in IoU on the validation set of cast iron parts has to be related to the reduced data quality. Further evaluating the deep learning method which we fine-tuned on the CT scans of low data quality (see Section 6.1), we see that the results already get significantly better. However, fine-tuning the deep learning method for a specific scenario should in general yield much more satisfying results. Therefore, we create some additional training data which more accurately resembles the data distribution of the CT scans of cast iron parts in the validation set. We simulate 50 additional CT scans using the same parameters for the virtual CT setup as for the iron cast validation set and again fine-tune the original model with a reduced learning rate of $1e^{-5}$ for 25 000 iterations. As expected, we can further improve the results even though we do not meet the high detection rate which the deep learning method achieves on the cast aluminum parts, which is due to the artifact-level of the cast iron CT scans being significantly higher than the artifact-level of the cast aluminum CT scans.

Lastly, in order to see whether the results of our simulations correspond to those in the real world, we provide a qualitative evaluation of a real CT scan of a cast iron part (see Figure 6.7c). While our original model could not cope with the reduced spatial resolution, under-estimating the size of the defects, we see that after fine-tuning on simulated CT scans of cast iron parts the deep learning



(a) IoU for different validation sets

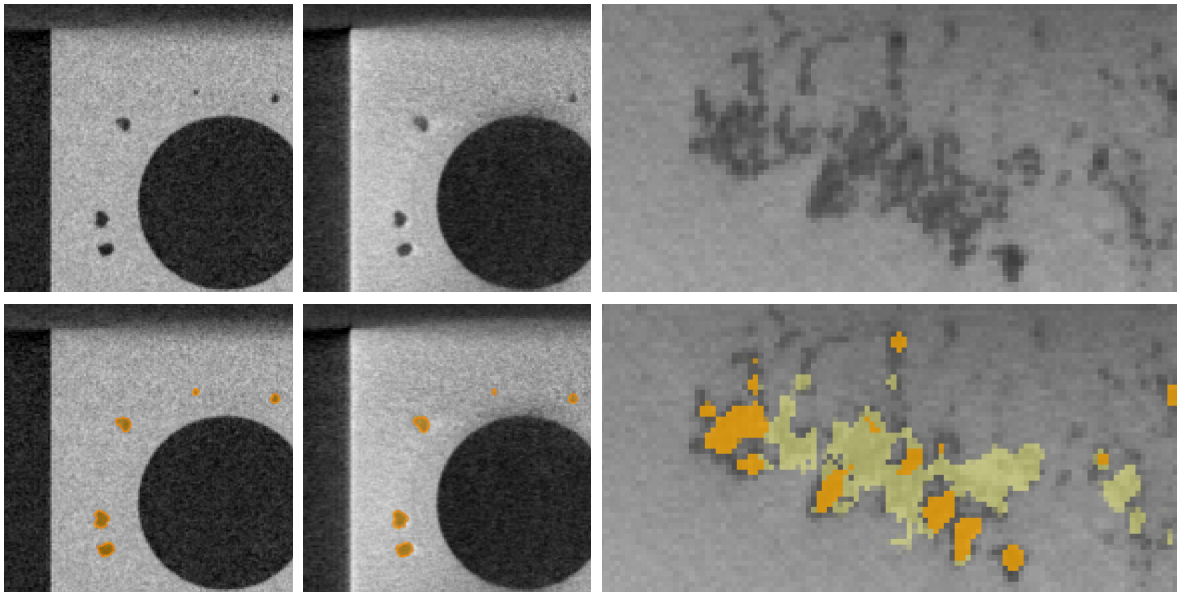
(b) POD for different validation sets

(c) IoU for fine-tuned models on iron cast data

(d) POD for fine-tuned models on iron cast data

Figure 6.6.: The plots in the top row show how the original deep learning model (dotted lines) performs on the different validation sets of different materials: aluminum casting (blue), plastic injection molding (green), and iron casting (red). We clearly see that the results on the iron casting validation set lacks behind the results on the other validation sets which is because of the high artifact-affliction of these CT scans. The plots in the bottom row show how fine-tuning helps to improve the results of the deep learning model on the challenging iron casting data. First, we fine-tune the model using the challenging low data quality CT scans of cast aluminum parts (see Section 6.1.2 or Table A.3) which already significantly improves the results (dashed lines). The fine-tuned model is capable of dealing with most of the artifacts occurring in the CT scans. Fine-tuning on CT scans which directly match the target (CT scans of cast iron parts, see Table A.5), however, yields even better results (solid lines). Explicitly tailoring the model towards a specific objective using a matching training set consequently leads to the best performance.

method also reliably detects and precisely segments the defects in real CT scans of cast iron parts. However, due to the reduced spatial resolution of the CT scan many of the smaller defects in the data set are blurred so that they still are not detected as individual defects but categorized as structural loosening.



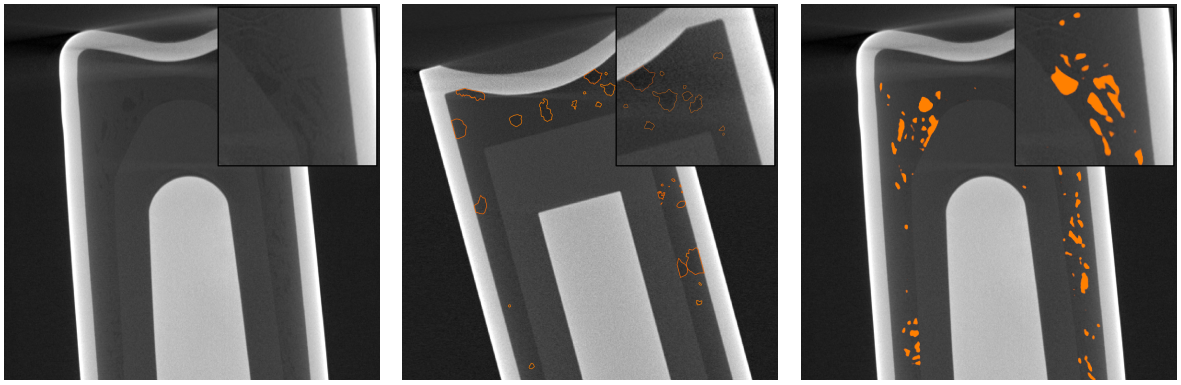
(a) injection molding (b) iron casting (c) results on a real CT scan of a cast iron part

Figure 6.7.: From left to right we show an example of a part taken from our injection molding validation set (using polyoxymethylene), and the same part simulated as iron cast using the iron alloy EN-GJL-150 [219]. The image on the right shows an example of a real CT scan of an iron cast bearing. The orange outline shows the ground truth from the defect mesh, the orange overlay shows the prediction output of the original deep learning method (a) and the fine-tuned iron cast version (b, c), respectively. In (c) we additionally show the predictions of structural loosening as yellow overlay. These cover most of the tiny defects which are blurred due to the reduced spatial resolution. The raw gray-value data is shown in the top row.

6.2.2. Assembled Multi-Material Parts

The inspection becomes especially difficult when we have to deal with assembled multi-material parts which consist of at least two materials which widely differ in their penetrability. For example, in Figure 6.8a we show the CT scan of a thermistor, a thermally sensitive resistor. Here, we need to determine the total defect volume in the low absorbing wax of the assembled part, which, however, is surrounded by a highly absorbing layer of copper, in order to ensure its reliability. It is particularly hard to achieve a sufficient contrast resolution in the wax part: On the one hand we need high energies to penetrate the hull of the part and on the other hand we cannot use arbitrarily high energies as the wax needs to have a chance to absorb at least some portions of the radiation. This not only limits the contrast resolution in the inner parts but also increases the artifact-level.

The deep learning method which is not yet fine-tuned has no chance of finding anything here. The target domain differs too widely from any known CT scan in our training set of cast aluminum parts. Thus, we replicate the object and simulate another CT scan which contains all the new challenges and for which we have the precise ground truth. The outer hull has a diameter of 8.0 mm, is 15.0 mm long, and has a thickness of 0.6 mm. The inner metal cylinder has a diameter of 3.0 mm, surrounded by about 2.0 mm of plastic. The rest is filled with our target material, i. e. wax. In this part we have to place our virtual defects. This time we only place large cavities and cracks, as the real scan does not contain any gas pores either. The new training set comprises three additional CT scans of dif-



(a) original thermistor (b) virtual counterpart (c) results of the fine-tuned model

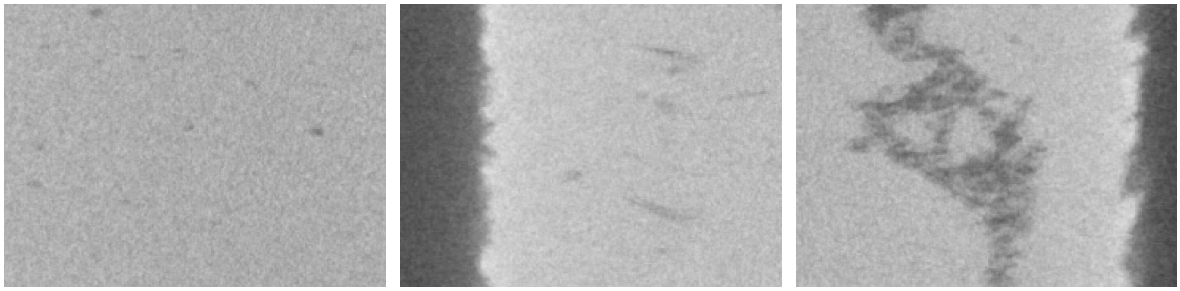
Figure 6.8.: We examine a multi-material data set were the objective is to find voids in a low-absorbing material which is surrounded by dense, highly absorbing metal hull (a). Thus, we have to deal with a very low contrast and strong image artifacts. The original model which is trained on aluminum cast cannot be applied here. However, it can serve as basis for fine-tuning using a simulated replica (b) with precise ground truth (orange outlines). After fine-tuning the model to this specific problem, we are able to obtain the desired results (c) on this more complicated task (orange overlay).

ferent exposure time, i. e. different noise-levels. Figure 6.8b shows an example patch taken from this training set. Then, we use another 25 000 iterations with a reduced learning rate of $1e^{-5}$ to fine-tune our deep learning method for this specific problem. Here, we find it beneficial to fine-tune all layers, not only the final layers which are responsible for the decision. Most probably this is because the new target domain moves too far from the original target domain of cast aluminum parts.

The real object and its simulated counterpart still show some deviations. Nevertheless, the results which the model achieves after fine-tuning are quite promising (see Figure 6.8c). Therefore, we argue that with the help of our simulation pipeline it is no problem to adapt the deep learning method to a new input domain—at least as long as the objective, i. e. the type and shape of the defects, does not change too widely. Having a sufficiently accurate virtual replica of the part under examination and the utilized CT setup at hand, we can tailor our deep learning models to achieve an unprecedented segmentation of defects in CT scans of any (scanable) part. Furthermore, we note that only very few additional CT scans are necessary for fine-tuning.

6.2.3. Additive Manufacturing

So far, we only considered changes in the data quality of the CT scans as a possible change of the input domain. We did not yet examine what happens if, for instance, a new type of defect occurs that the model has never seen before. To provide a qualitative example, we have a look at additively manufactured metal parts. Additive manufacturing (AM) is a upcoming manufacturing technique which promises to build stable and lightweight parts which, for example, are used in aerospace industries. These parts are printed layer by layer. A laser (or electron beam) heats up the material in the powder bed until the particles are molten and connected. When a layer is finished, a new layer of powder is laid on top. The quality of the printed part depends on many parameters like the energy input of the laser, the path of the laser, i. e. the distance of its tracks, the quality of the powder, or the

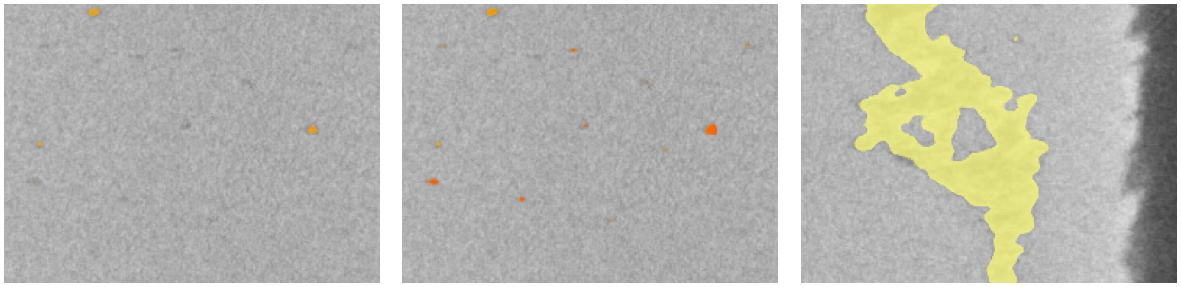


(a) small and scattered pores (b) tiny delaminations between two layers (c) cavities with enclosed residual powder

Figure 6.9.: When examining additively manufactured parts, we are confronted with new types of defects, which widely differ from everything seen in cast metal parts.

height of the current layer of powder. Respectively, it happens that underlying layers are molten up again, that the current layer is molten but not connected to the next layer, or that the powder is not molten at all. This can lead to several types of defects, which can significantly differ from those we observe in cast metal parts. With the energy input being too high, the material in previous layers are molten multiple times and, hence, are again exposed to the surrounding mix of air and protective gas. This and the very high local temperatures contribute to the formation of gas pores. However, in contrast to metal casting, these pores are much smaller and widely distributed throughout the entire part (see Figure 6.9a). These pores often have a diameter of only a few micro meters. If the energy input is just high enough to melt the topmost particles in the powder bed it might happen that the layer which is currently printed is not entirely connected to its underlying layer. These so called delaminations occur as small and very thin voids in the CT scan (see Figure 6.9b). In case of an insufficient energy input in multiple consecutive layers, it further can happen that there occur cavities which include residual powder. The residual powder was either never or only partially molten (see Figure 6.9c). For further information on different AM techniques, the importance of different production parameters, and defect reduction techniques, we refer to [220]. In addition to the novel types of defects, AM often uses more dense and solid materials which bring along further difficulties for the acquisition of CT scans. On the other hand, the cycle time for the production is much longer, which at least allows for long scan and analysis times.

Our deep learning method properly detects the larger instances of the gas pores (see Figure 6.10a) and further manages to identify the cavities with included residual powder as structural loosening (see Figure 6.10c). However, the model struggles with the detection of the smaller instances of the gas pores. Despite having small defects with a diameter of 3 voxels or less in our training set, not even the deep learning method detects these defects reliably. A reason might be, that the IoU (and, hence, the dice loss, too) favors the detection of larger instances (see Section 5.3.3). Thus, we re-train our model mixing in a second loss function: the focal loss [221]. This additional loss function puts more weight on particularly hard examples by lowering the contribution of the vast majority of easy examples of correctly identified voxels, e. g. from the background, to the total loss. Equation (6.1) shows the definition of the focal loss, where we choose $\alpha_{\text{inner defects}} = 0.25$ and $\gamma = 2.0$ as suggested by the authors. The parameter γ determines the rate at which the loss contributions are down-weighted when the prediction becomes more confident. The balancing parameter α_c puts further weight on the target class—the inner defects. The total loss function for the re-training is then given by the sum of both losses: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Tversky}} + \mathcal{L}_{\text{focal}}$.



(a) (b) (c)

Figure 6.10.: With our deep learning method we are able to find and segment the larger instances of the gas pores ((a), orange overlay). If we combine the prediction of the original model and the prediction of the model we trained to detect small defects in particular using the iIoU as loss function, we are able to lower the minimum size requirements for a defect to be detectable ((b), dark orange overlay). The raw data of (a) and (b) is shown in Figure 6.9a. The voids with inclusions of residual powder are detected as structural loosening ((c), yellow overlay). The raw data of (c) is shown in Figure 6.9c. Unfortunately, the delaminations, shown in Figure 6.9b, remain undetected most of the time and, therefore, are not shown here.

$$\mathcal{L}_{\text{focal}} = -\alpha_c(1 - p_c)^{\gamma} \log p_c \tag{6.1}$$

Nevertheless, the detection of small defects remains difficult, the loss does not emphasize smaller instances enough. Our second approach is to re-train the model putting a special focus on smaller instances in particular by turning the iIoU metric into a loss function. Instead of down-weighting easily classified voxels, we weight the voxels according to the size of the defect instance they belong to. Thus, the adjusted loss function puts a weight on the dice loss depending on the size of the considered instance compared to the mean size of all instances and is computed as $\mathcal{L} = 1 - \text{iIoU}$. The mean defect size is pre-computed over the entire training set so that each defect voxel can be assigned the weight of the size of the instance it belongs to. Therefore, the weight assignments are independent of whether the training patch of the current iteration shows the complete defect or only a part of it. This loss function further implicates that larger instances receive lower weights, which results in the segmentation of larger instances becoming less precise. Having a look at the POD confirms this behavior: We properly detect smaller instances but significantly lose detection probability for larger defects as the segmentation becomes less precise. For the largest defects, the POD recovers as we have less boundary voxels compared to inner voxels, which are properly detected. To obtain a good detection of small gas pores and a precise segmentation of larger instances, we combine the results of both models (see Figure 6.10b). This further means that the prediction time doubles, which, however, does not carry that much weight compared to the long production cycle times.

The delaminations, however, remain undetected most of the time. Therefore, we need to properly model all type of defects which we like to detect. This makes further adaptations necessary. In the next section we demonstrate that adapting the simulation pipeline to not only vary the data quality of the CT scans but to also adjust the simulated defect types to a given scenario is the key to a reliable and fast inspection.

6.2.4. Crack Detection in Nickel-based Alloys

New types of defects are not only introduced by new manufacturing techniques. In cast metal parts it is possible that large splits or fractures emerge after straining the part—especially when examining the results of fatigue tests [150]. It is important to detect these cracks as they have a huge impact on the stability of the part. In terms of the CT data, they look similar to the delaminations we observe in AM (see Section 6.2.3). The main difference is that the splits occurring during fatigue testing can be significantly larger. Therefore, we now change the objective: instead of only looking for gas pores and shrinkage cavities we are also interested in very thin and widely ramified fractures which meander throughout the entire part. We again provide a qualitative example: We examine five tension rods made of a nickel-based alloy, which, in addition to the new type of defect, impedes the inspection via CT due to its high absorption coefficient. These tension rods were scanned with 200 kV at 0.3 mA and a filter of 1.0 mm of copper. Each CT scan comprises 1620 projections with an exposure time of 1.0 s per projection.

As expected, our deep learning method fails to properly detect the fractures in the data set. The low data quality of the CT scans, which is inevitable due to the high density of the material, additionally impedes the inspection. Our model which we fine-tuned for challenging CT scans at least finds the cavities which partly function as initiation points for the fractures but not the fractures itself. The fractures are only a few voxels thin and appear very blurry due to the high energies and the large focal spot sizes associated with them which we have to use to acquire the CT scans. As before, when considering the delaminations, not even the training for small defect instances, which reduces the minimum required size of a defect to be detectable, did help. Therefore, we need to revisit the simulation pipeline to add the new target defect type to our training data. Here, we bring the large splits into play (see Section 3.3.1) to train the model to detect these fatigue fractures as well. Figure 6.11 shows a few example slices of the additional training data. We use the same scan parameters as we used for the additional training data for in-line scenarios so that we not only detect the fractures but also cope with the low data quality of our test CT scans. When creating the precise labels from the defect meshes we need to consider that the large splits might overlap with other defects as the creation of the splits is independent from the creation of the other defects. We do not remove overlapping inner defects, to simulate occurrences where an inner defect functions as starting point of a fracture. However, this needs to be considered in the training labels. Thus, an inner defect always has higher priority than a large split. Because this is a completely new type of defect which differs from the present defects, we decide to simulate a complete training set of 675 CT scans to obtain as most variety of data as possible. Furthermore, we train the model from scratch so that the filters of the early layers can better adapt to the new type of defect as well. We train the model for 150 000 iterations, the same way we trained the original deep learning model for cast aluminum parts.

In addition to the training data, we simulate 10 more CT scans as validation set to precisely determine the segmentation quality for the new defect type. The deep learning method achieves an IoU of 68.9 % for the detection of the thin and large splits (82.3 % in the CT scan with the best data quality and 51.2 % in the CT scan with the highest artifact level). We see that the data quality has a much higher influence here due to the limited extent of the defect structure. With the new model being explicitly trained to detect thin and flat structures, we detect the fatigue fractures in the CT scans of the tension rods (see Figure 6.12a). As the labels properly distinguish between the defect types “large splits” and “other inner defects”, we also obtain this distinction during inference (see Figure 6.12b). This enables

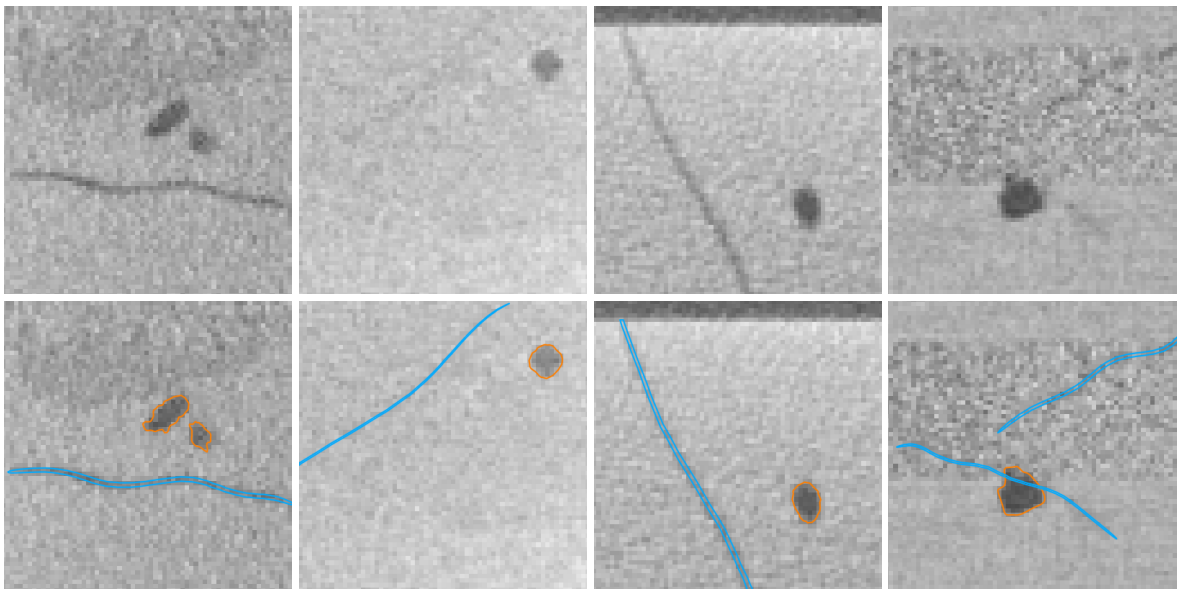


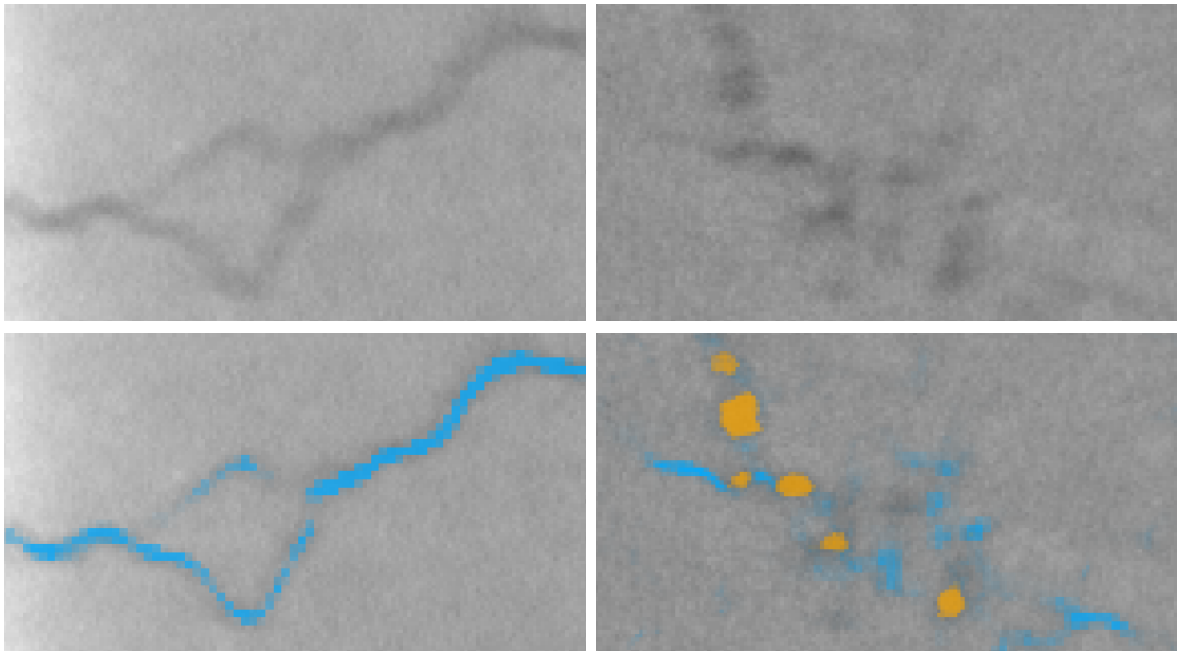
Figure 6.11.: Additional training data for crack detection: We use the large splits which we presented in Section 3.3.1 to prepare our deep learning model for the detection of large cracks arising from the fatigue of the part. The large splits are indicated by the blue outline, other defects by the orange outline, respectively. The images in the upper row show the raw gray-value data. We see that the training set also includes splits which have a diameter of less than a voxel. Here, we can only guess that there is a defect by taking a large context into account. As the large splits are placed independently of other defects, we need to take care of overlaps when creating the per-voxel annotations. The other defects always beat the large split defects.

us to spot the cavities which serve as initiation point for the formation of fatigue fractures.

Having seen all these results, we conclude that our fully automated simulation pipeline in combination with deep learning methods is a mighty toolset promising to solve many particularly difficult NDT problems. Most of all, the training of deep learning models benefits from the precise labels we get from the simulation. This further allows to aim for the segmentation of especially small structures. The only requirement is that we have a virtual model which is as close to the target as possible. However, it is not necessarily adverse to make some approximations as long as the overall training data is close enough to reality. To get even closer to reality, we could, for example, apply a finite elements casting simulation to identify porosity hot spots in a given CAD file and use this information to create an increased number of defects at this precise location. Other fields of application could be to help with the segmentation of short glass fibers in fiber composite materials [46], the detection of individual parts in large loose piles, or even solving custom problems in more complex in-line inspection tasks.

6.3. Distinguishing Gas Pores From Shrinkage Cavities

One reason for the necessity of having a precise segmentation of the casting defects in a CT scan is that it provides the prerequisites to determine the type of the defect. Due to their smooth and



(a) (b)

Figure 6.12.: The original model would not detect the large cracks that meander throughout the tension rod. But, by retraining the model with the large splits of the new training data, the model does. In fact, we are also able to distinguish the fatigue fractures from other defects like the cavities they originate from. These real CT scans are heavily artifact-afflicted because the nickel alloy is highly absorbing. Therefore, we have to deal with severe cupping artifacts arising from heavy beam hardening. The blue overlay shows the prediction result for large splits, the orange overlay the prediction result for inner defects.

clean surface, gas pores contribute less to the initiation of fatigue fractures and, therefore, are often considered less harmful than, for example, shrinkage cavities—at least up to a certain amount and depending on their position in the object [1]. We model our virtual casting defects to resemble each type of defect as good as possible. Thus, instead of using the precise segmentation for a subsequent rule-based classification, we can utilize the information of the defect meshes to train our deep learning method to directly predict a classification. Instead of separating the precise per-voxel ground truth in inner defects and surface defects, we separate it in the different defect categories: gas pores, shrinkage cavities, and solidification cracks (see Figure 6.13).

Larger instances are easier to categorize than smaller instances. It is easier to determine whether the defect is more spherical with a smooth surface or more of a branching structure with a rugged surface. In contrast, for smaller instances these differences become more and more subtle as the resolution of the voxel grid sets the limits here. Having a defect that consists of only very few voxels, it is very hard to say if it is a micro shrinkage cavity or a small gas pore. We would need to distinguish the defect type on basis of a slightly varying gray-value which can also arise from noise. As our training set also contains very tiny defects, training on all defects does not seem expedient. Hence, we evaluate the minimum size which is necessary for a proper classification by training different classification models with an increasing minimum size requirement. During training, the labels of defects which are smaller than the minimum size requirement are ignored, i. e. weighted by zero. We train four models with the minimum size requirement set to an ESD of 5, 7, 9, and 11

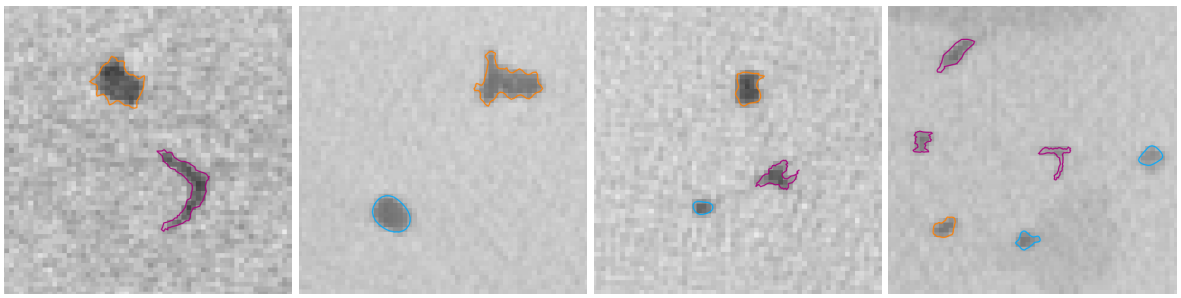


Figure 6.13.: Some examples of the training data with separated inner defects: gas pores (blue), shrinkage cavities (orange), and solidification cracks (purple). While the larger instances in the first two examples are relatively easy to distinguish, the smaller the defects become, the less distinct is their voxel pattern.

voxels, respectively. Furthermore, we train a fifth model without any size constraints.

To provide our model with sufficient information about the entire defect, we further have to slightly modify our defect segmentation architecture (refer to Figure 4.14): Instead of passing the segmentation of the intermediate layers into the refinement step, we create an *intermediate feature representation* of 16 features and concatenate those features before passing them to the refinement step. Now, the input of the refinement step is a volume of 48 feature channels. The deep supervision is realized by three additional convolutional layers, which are trained to create a segmentation mask from the respective intermediate feature representations. However, the additional features and convolutions not only increase the capability of the model but also have an impact on the memory footprint and the required computation time. About 650 s are necessary to process a volume of $1000 \times 1000 \times 1000$ voxels now.

As we train on defects of different sizes, it is important to also take into account the size of the defect instances during validation. Thus, we evaluate the models in terms of their iIoU instead of using the plain IoU measure. In Figure 6.14 we show the iIoU as a function of the minimum size requirement. The main results of the evaluation of these five models on our simulated validation set are twofold: First, we notice that the overall classification performance is rather bad—independent of the defect size. Second, the mean iIoU over all classes is best for a minimum size of 7 voxels. The results for gas pores becomes even better for a minimum size of 9 voxels, but the detection rate of the other defect types drops significantly. In consequence, we stick with the model which has a minimum size requirement of 7 voxels and further examine its results.

In Figure 6.15 we show some results of the model with a minimum size requirement of 7 voxels taken from our simulated training set. We see that for unambiguous instances the model has no problem to assign a proper category. However, not all instances are that distinct: There are gas pores with a slightly dented hull tending to look more like a shrinkage cavity. The other way round there are shrinkage cavities which are more roundish and look just like gas pores. Furthermore, some instances of solidification cracks are more compact and, thus, are more likely to be confused with shrinkage cavities. The model assigns (at least partially) the wrong defect type to these instances.

Therefore, the results we obtain for real CT scans are less trustworthy, too. In the CT scan of the aluminum part we find two larger defect instances which our classification model identifies as shrinkage cavities, the smaller defect instances are assigned the label of a solidification crack (see Figure 6.16a). The smaller gas pores are too small to be classified, i. e. smaller than the minimum size

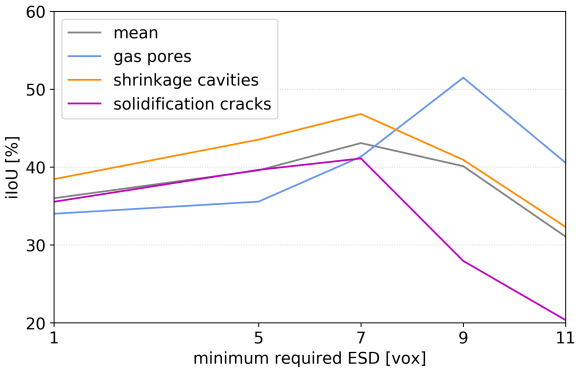


Figure 6.14.: The iIoU as a function of the minimum size requirement used for training. Although we exclude the smaller instances so that the network can focus on the decisive aspects of the defects, the overall result of none of the models is really overwhelming. However, we see that the mean iIoU culminates for a minimum size requirement of 7 voxels. For a larger minimum size requirement the model indeed better identifies gas pores but struggles with the other defect types. We have too few defect instances to train on and too many possibilities of different defect shapes.

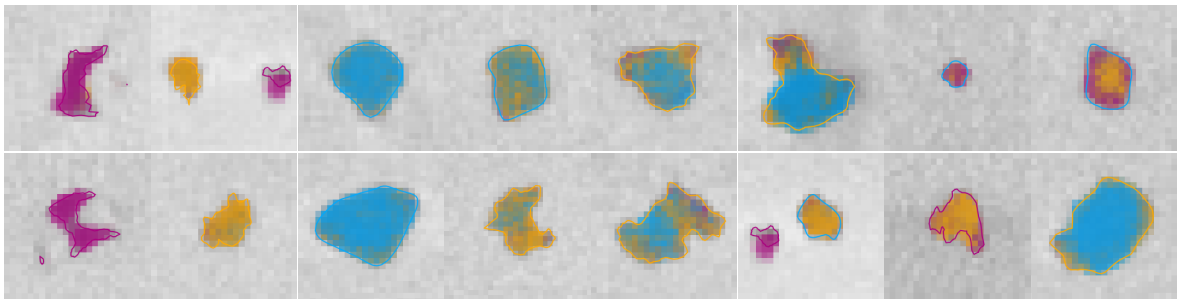
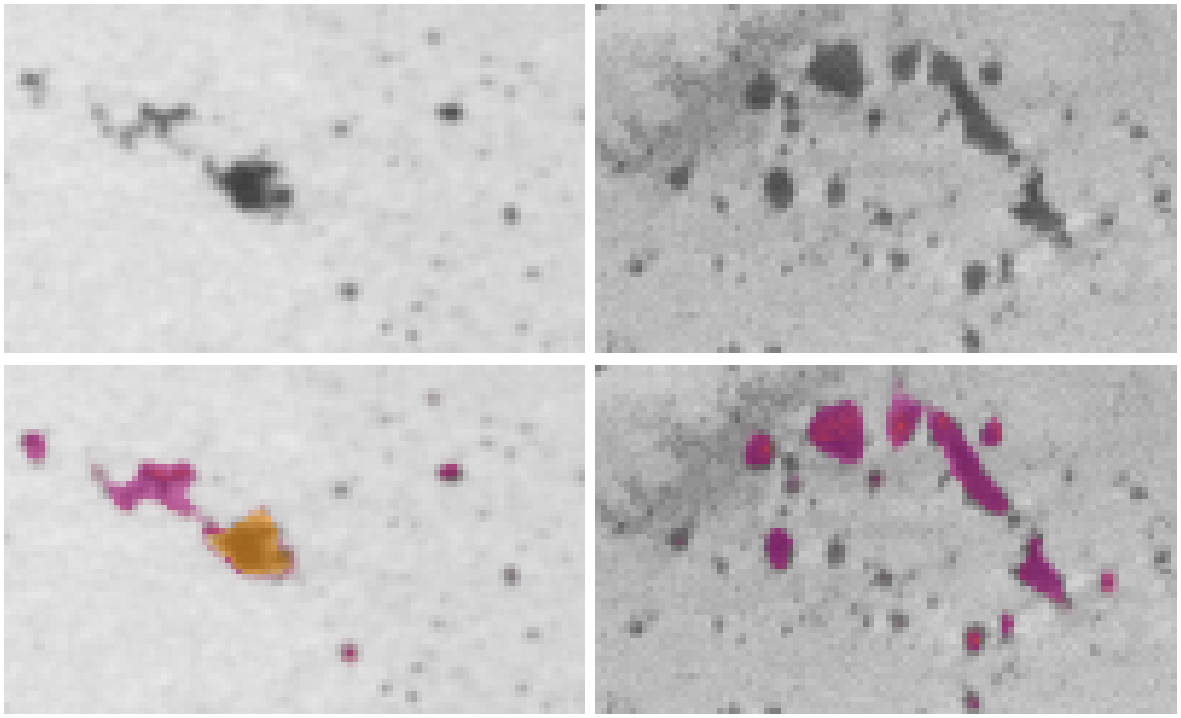


Figure 6.15.: Some results of our classification model. From left to right the prediction becomes worse. The results are taken from our simulated validation data. The differences between gas pores and shrinkage cavities as well as between shrinkage cavities and solidification cracks is not always clear; it is more of a smooth transition. The overlays show gas pores (blue), shrinkage cavities (orange), and solidification cracks (purple). The outline represents the ground truth from our defect meshes and the overlay the prediction of our classification model.

requirement of 7 voxels. Our experts agree with the decision of our model about the shrinkage cavities as they clearly show a rugged surface. In the CT scan of the pivot cap our classification model labels most of the instances as solidification cracks (see Figure 6.16b). Our experts, in contrast, would clearly categorize them as shrinkage cavity. At least, the model also correctly identified two larger gas pores (which are not shown in the image).

All in all, the results are to some extent correct and definitively point into the right direction. However, the instances of the different defect types in our training set are sometimes too indistinct—even though we carefully designed them in accordance to current standards and guidelines. This complicates the prediction on our validation data and also affects the results on real CT scans. Therefore, the current training set is good enough to reliable detect and precisely segment defects in CT scans of cast aluminum parts but not to categorize them. We have already shown in Section 6.2.4 that it is possible to use simulated data to train a stable distinction between fatigue fractures and other defects. Hence, to enable a better distinction between gas pores, shrinkage cavities, and solidification



(a) In the CT scan of the aluminum part the identification of the shrinkage cavity seems to be right and the solidification crack is also plausible.

(b) The solidification cracks that the model identified in the CT scan of the pivot cap are more likely to be shrinkage cavities.

Figure 6.16.: Examples for results of the defect classification model on the CT scans of the pivot cap and the aluminum part. The overlays show the results of the classification model: shrinkage cavities (orange) and solidification cracks (purple). The smaller defect instances are smaller than the minimum required size of 7 voxels and, therefore, not labeled.

cracks we will have to revisit the mesh generation step in our simulation pipeline.

6.4. Model Uncertainty

Up to now the major concerns of domain experts are twofold: "what happens on CT scans which the model has never seen before?" and "can we detect when the data quality of our CT scan becomes too poor?" These concerns emerge from the tendency of deep neural networks to output false positive or false negative results with a high probability even for unseen and *out-of-distribution (OOD) data*. During training the output nodes of a neural network are trained to yield large positive values if the input corresponds to the class represented by a node or large negative values otherwise. These values are then mapped to the value range of $[0; 1]$ over all classes using the softmax function, in order to form some kind of pseudo probability. Therefore, we may not confuse the model output with its confidence, meaning deep neural networks being overconfident on their decision [222, 223].

In the field of NDT, and the segmentation of defects in particular, the definition of OOD data is not as distinct as showing letters to a model which trained to distinguish between numbers. It is more of a continuous nature: When training our model on CT scans of cast aluminum parts only, a CT scan

of a cast iron part already is OOD—even though the results of the model might still be convincing. A CT scan of a multi-material part even further deviates from our original training distribution. Furthermore, there are more subtle forms of OOD data. For example, if the source becomes weaker over time, the detector becomes less sensitive, or we simply reduce the exposure time, the resulting CT scan can also be considered as OOD due to its increased artifact-level. Depending on the distance to the training distribution the quality and precision of the results will degrade, however, not the "confidence" at which our model outputs them. In the extreme, we still obtain high probability results for both kinds of OOD data but would not detect a single defect.

These two concerns and the two respective scenarios they are related to, basically tackle the two types of uncertainty which are involved when training a machine learning model: the *epistemic uncertainty* and the *aleatory uncertainty* [224]. The epistemic uncertainty describes the uncertainty of the model on data which is not covered by the training set, for example, when moving from cast aluminum to cast iron and multi-material parts. The aleatory uncertainty describes the uncertainty under the presence of noisy input data, for example, due to the fatigue of individual components of the scanner or simply when reducing the exposure time. While we can reduce the epistemic uncertainty by adding more training data, the aleatory uncertainty cannot be reduced by adding more data [225].

The main question is, how can we estimate the uncertainty and can so detect OOD data, on which the results of our model are less reliable? Deep neural networks do not provide any measure of uncertainty—at least not by default. Therefore, we examine four frameworks which augment deep neural networks with some kind of uncertainty. To measure the uncertainty we need a probability distribution over the output. Our neural network, however, only provides us with one sample of this distribution and as long as we do not change anything we always will obtain the same sample. Therefore, we somehow need to introduce changes to the process to obtain multiple samples. Then, we can compare the different outputs and derive an uncertainty measurement from their deviation [222, 226–228]. This is the approach of the first three frameworks we examine. The fourth approach is to train the model to not only predict a segmentation result but to further output a score which represents the confidence at which this decision was made. With that we can add a notion of how confident the results are and should be able to detect whether the output becomes less trustworthy and whether the need for a second opinion, e. g. by a human inspector, arises. We test these methods on our simulated validation data as well as our real CT scans of the aluminum part and the pivot cap. As OOD data we choose the CT scan of the pivot cap which is reconstructed using only 110 projections (compare Section 6.1) and the CT scan of the multi-material wax part (compare Section 6.2.2).

Monte-Carlo dropout This approach uses dropout during the inference to approximate a Gaussian process [222]. What helps during training to avoid overfitting, helps to estimate the uncertainty of a model during inference. The basic idea is to obtain different outputs, i. e. different samples of the posterior distribution, by randomly discarding a portion of input. This way, the model only sees different portions of the total information during each forward pass and, thus, produces M slightly varying results in M forward passes. With an increasing distance of the input data to the training distribution, the differences between the results should increase. Furthermore, with $M \longrightarrow \infty$ we would get the same results as we would get from the model without dropout [222]. As each of the M forward passes makes use of a different combination of neurons in the model, it can be seen as asking an ensemble of M different models. During

inference we use a reduced dropout rate of only 10 %, which is sufficient to simulate an ensemble but still yields results of high precision. As we need M forward passes (at least starting at the first dropout layer) the inference time is increased by the factor of M.

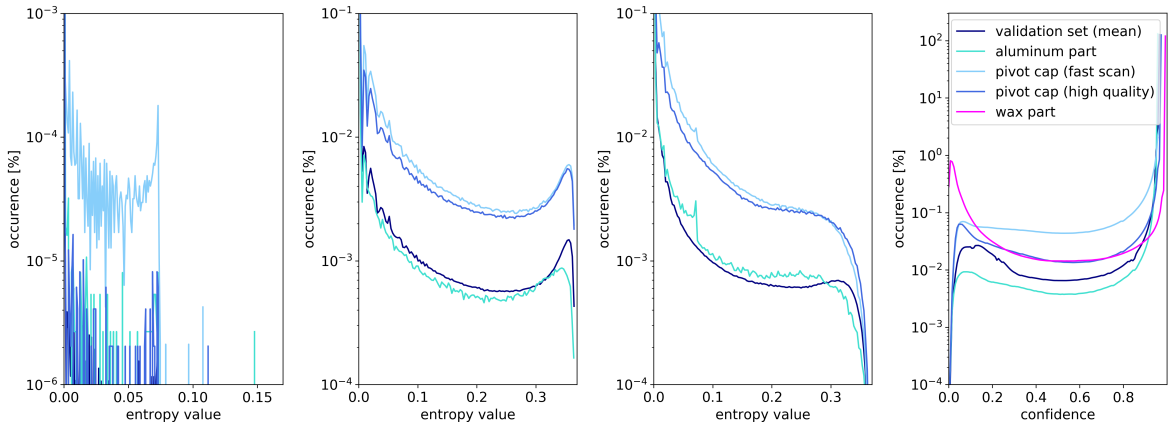
Conditional training Another way to simulate an ensemble is to vary the input of the model [229, 230]. Per data point of the input, the conditional variational auto-encoder additionally gets a n-dimensional conditional vector z as input [230]. While this method is originally used in image-to-image translation to model a multi-modal output, i. e. to generate various different output images, we employ this method to introduce slight differences in the segmentation results, which become more significant as the input data yields less distinct clues, i. e. moves away from the training distribution. Instead of encoding a noisy version of the desired output in z, we draw z (with $n = 8$) from a standard normal distribution. Then, we concatenate z to the input [230]. We do this M times for M different z. Therefore, the prediction time is again increased by the factor of M. In contrast to the Monte-Carlo dropout, however, the model has the complete input information available in each forward pass.

Deep ensemble Instead of simulating an ensemble using dropout or a conditional input, we train a real ensemble of deep neural networks comprising M models. The deep ensemble requires the M models to be trained with a different random initialization [228]. These models will each end up in a different local minimum and, therefore, yield slightly different predictions. For this approach we again have an increased prediction time as we need to perform one forward pass for each of the M different models. Consequently, the training time is increased by a factor of M and we have to store M times more weights.

Predicted confidence The final approach is to train the model to directly estimate the confidence of its prediction [231]. For this, we add a separate output branch to the model, which represents the estimated confidence c. During training the model can request help by means of the ground truth. Depending on the output of c we mix the prediction p with the ground truth y as guidance: $p' = c \cdot p + (1-c) \cdot y$ [231]. To avoid the network becoming lazy and always requesting the full help by predicting $c = 0$, c is penalized by an additional loss $\mathcal{L}_c = -\lambda \log(c)$ [231], i. e. requesting help costs. The weight λ of this additional loss function, however, is not fixed but varies over the time. An additional "budged hyperparameter β" [231] is introduced for training. In each iteration λ is increased by 1 % if $\mathcal{L}_c > \beta$, i. e. the help becomes more expensive when the budget is exceeded, or decreased by 1 % otherwise. We choose $\beta = 0.01$ which makes the request of help more expensive in general. The huge benefit of this approach is that we only need a single model and a single forward pass to obtain a segmentation as well as an estimate of the confidence.

For the experimental evaluation of detecting OOD data and measuring the uncertainty of the model, we fall back to the standard U-Net because training and inference (along a single axis) are much faster. Furthermore, we restrict the models to only predict two classes: defective or not. However, we train the model on slabs which comprise three adjacent slices to avoid severe inconsistencies in the results. Each model is trained for 150 000 iterations with a learning rate of $1e^{-4}$.

Except for the last model, where we directly obtain an uncertainty measure as output, we compute the predictive *entropy* $H = 1/M \sum p_m \cdot \log(p_m)$ [232] over $M = 5$ predictions to capture their differences and to get an indication of the uncertainty of the model. As a baseline we compute the mean entropy of the predictions from the CT scans in our validation set. Then, we compute the entropy



(a) Monte-Carlo dropout (b) conditional training (c) deep ensemble (d) predicted confidence

Figure 6.17.: The histogram of the entropy of the aluminum part, a fast CT scan of the pivot cap, and a CT scan of good data quality of the pivot cap compared to the mean entropy of the realistically simulated training set for the different methods: (a) the Monte-Carlo dropout, (b) the conditional training, (c) the deep ensemble. Figure (d) shows the corresponding histograms of the predicted confidence. Furthermore, it includes the results of the wax part (for the other methods the entropy level for the wax part was close to zero).

of the predictions which we get on the CT scan of the aluminum part and the CT scan of the pivot cap with the best data quality, i. e. the full set of projections. Both are cast aluminum parts and are scanned similarly to our training set. Therefore, we expect the entropy to be somewhat close to our baseline. Furthermore, we compare the entropy on the CT scan of the wax part and the fast CT scan of the pivot cap, which is reconstructed from only 110 projections to our base line. These CT scans are OOD data: The first is a multi-material data set with an unseen gray-value distribution, the latter has a significantly increased artifact-level. Hence, we expect the entropy to be significantly larger than in our baseline.

In Figure 6.17 we show the histograms of the entropy (and the predicted confidence) of the different approaches. First of all we notice that the entropy histograms of the Monte-Carlo dropout approach are less meaningful. The predictions of the different runs are quite consistent. Only for the fast CT scan of the pivot cap the entropy slightly increases (see Figure 6.17a). For the other methods we see that even on the validation set, which is drawn from the same data distribution as the training data, we obtain a certain entropy level. The entropy of the results of the CT scan of the aluminum part ranges at about the same level. The entropy of the results of the CT scan of the pivot cap with high quality, however, significantly lies above our baseline. In contrast, the fast CT scan of the pivot cap, which we would expect to deviate farther from the baseline and the CT scan with the better data quality, lies relatively close to its high quality counterpart (see Figures Figure 6.17b and Figure 6.17c). Then again, from Section 6.1 we know that the results of the CT scans of varying image quality do not differ by much, which explains why they also show about the same level of entropy. For the entropy-based methods we, finally, see that there is almost no uncertainty in the prediction for the wax part. This is because of all predictions agreeing that there is no defect in the data.

To better understand the results in Figure 6.17a–c we need to have a look at where the uncertainty arises from. We see that the differences in the predictions—and, thus, the entropy—mostly occur in the border region of defects and close to defect-like artifacts like rings and streaks as well as

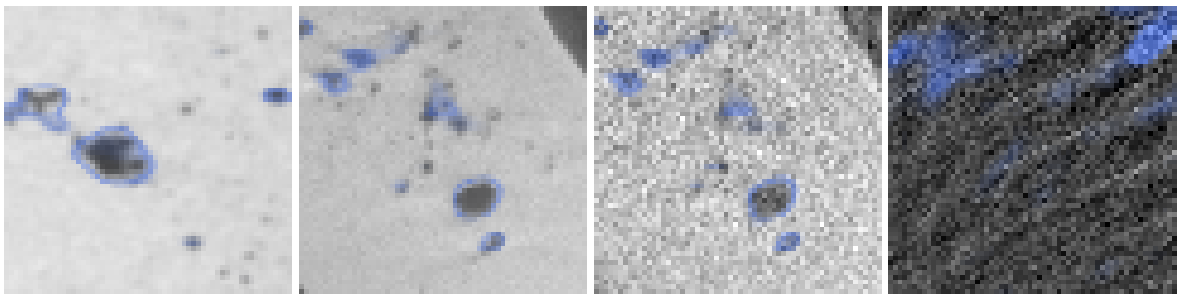


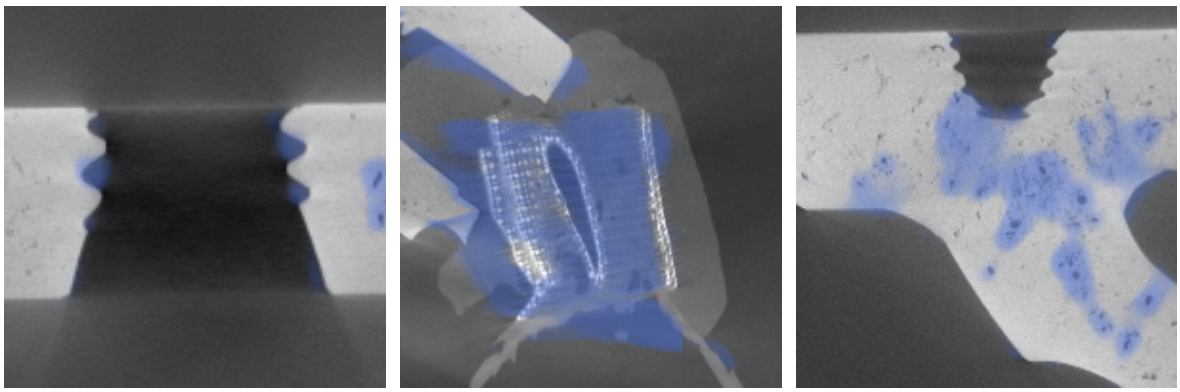
Figure 6.18.: The uncertainty in terms of the entropy becomes particularly visible in the border regions of defects and unprecedented defect-like artifacts like the aliasing artifacts that occur due to a radial under-sampling. The blue overlay shows the entropy of the deep ensemble method.

aliasing artifacts (see Figure 6.18). Even though we train with a precise ground truth, there is still some uncertainty about where a defects ends and where the material begins, which further supports the need for precise labels. Moreover, this means that we need to consider that the total amount of entropy in a CT scan of a cast aluminum part with less defects is in general lower than in a CT scan of a part with more defects. Therefore, we argue that an increase in entropy is not necessarily an indication of OOD data—at least not for the segmentation of defects in CT scans.

More promising is the predicted confidence: Here, we even obtain a useful signal for the wax part (see Figure 6.17d). Furthermore, the predicted confidence nicely indicates unfamiliar objects and portions of the part under examination in the CT scan which are not covered by the training set (see Figure 6.19). These indications contain screw threads (which we also recognized as challenging in Section 4.3.4), the mounts which are used to mount the parts upright in CT system, instances of structural loosening with a lot of mixed-in gas pores, and aliasing artifacts. This can be very useful when developing a new method or performing the first tests on novel data to see what requires more attention during modeling or training. However, the predicted confidence comes at a price. The segmentation results are significantly worse and the minimum diameter for a defect to be detectable becomes considerably larger which renders this method impractical. This is most probably due to the hint system acting as a strong regularizer, as stated by the authors of this method [231].

In general, we are able to detect an increase of uncertainty, in terms of the entropy of the predictions, when reducing the scan time, i. e. increasing the artifact-level, especially the noise. However, these differences are much smaller than the differences caused by defect-affliction. Furthermore, neither the longer prediction times nor the reduced segmentation quality are desirable, especially in in-line scenarios where an automated detection of a decrease in scan quality would be most useful. Therefore, we think it is more expedient to scan a specific test specimen every once in a while which allows to precisely measure the scan quality. As we showed in Section 5.2 the detectability of defects is heavily correlated with the data quality of the CT scan. A detected decrease in the data quality of the CT scan of the test specimen is very likely to impede the detection of defects in the CT scans of the actual parts as well.

In this chapter we examined the possibilities offered by the combination of our simulation pipeline and our deep learning method—and where the limitations of this duo lie. We find that (i) the robustness of the results and the high inference speed makes the deep learning method a good candidate for in-line inspection scenarios, where we meet especially challenging CT scans with a high



(a) screw threads (b) mount (c) structural loosening

Figure 6.19.: The predicted confidence nicely indicates where the model is struggling: The confusion of screw threads and surface defects (a), the mounts which are used to mount the part in the CT system (b), accumulations of many tiny gas pores and instances of structural loosening (c), and, of course, aliasing artifacts (not shown in the images but similar to Figure 6.18). The blue overlay shows the predicted confidence of the model.

artifact-affliction. (ii) The results of our deep learning method can be improved further by utilizing our simulation pipeline to create more task-specific training data and fine-tune the defect detection model on them. This is particularly beneficial for complicated inspection tasks which comprise CT scans of poor image quality due to physical limitations. (iii) While the pure segmentation results are convincing, when it comes to a classification of the detected defects, our method fails. The different defect types are too similar and require further refinement. Then, we probably will be able to not only correctly classify fatigue fractures but also are able to tell apart gas pores and shrinkage cavities. (iv) We could not further comfort the worries of the domain experts by building a framework that automatically detects OOD data. However, as discussed in Section 5.2.2 by computing standard quality measures like the MTF and the CDF it is possible to draw conclusions about the detectability of defects. Hence, we argue that this is not much of a loss.

7. Conclusion

We trained a deep learning model for the precise semantic segmentation of defects in CT scans of cast aluminum parts by only using realistically simulated data. In doing so, this thesis covered a complete way of exploring a novel field of application of deep learning methods, going from the acquisition of training data, over the creation of the model, to a profound evaluation. Furthermore, in this work we provided a broad exploration of the final model on a variety of different tasks. The results of this work are threefold:

Generating Meaningful Synthetic Training Data. First, we examined different approaches to obtain the training data. We made high-quality CT scans with a high spatial and contrast resolution for labeling and then transferred these labels to "normal" CT scans of the same part. However, the ambiguities of the experts' labels did not allow for a precise segmentation. Thus, we further evaluated three methods of obtaining synthetic data, for which a precise ground truth can be computed, namely domain randomization, generative models, and realistic simulations. We found the realistic simulation of CT scans suiting our task best as such training data, in addition, is explainable to domain experts. Hence, we developed a fully automated simulation pipeline to create the necessary amount of training data: First, we create virtual die casts with procedurally generated defect meshes placed on their inside using a force-directed graph drawing algorithm. Second, we realistically simulate the projections of a CT scan with a ray-casting method. Finally, we reconstruct the CT scans from the simulated projections and compute the per-voxel ground truth from the precisely aligned defect meshes. This pipeline enables us to create an arbitrary amount of data. For the purposes of this work we started with 675 CT scans totaling 1.35 TB and comprising 418 095 defects.

Challenging Deep Learning. Before building our deep learning method, which combines the encoder-decoder architecture and skip-connections of a U-Net with a refinement step to correct the high-recall intermediate outputs, we introduced a traditional machine learning method and a filter-based method as baseline. The filter-based method uses morphological filters to create a "defect-free" specimen out of a CT scan of an arbitrary part to then perform an anomaly detection by comparing the original part to the generated "defect-free" version. The traditional machine learning method extracts simple filter-based features for each voxel in the data set and uses a random forest to later classify each voxel as "defective" or not. We compared all three models in terms of their POD, which is a measure familiar to the domain experts and tells us at which probability a defect of a given size can be found in a CT scan of given data quality, and in terms of their IoU, a typical measure for semantic segmentation, which provides us with a score of the overall quality of the segmentation mask. While both machine learning methods perform equally well on CT scans of high to medium data quality, we found our deep learning method to outperform all other methods on CT scans of low data quality—on our simulated validation data as well as on real CT scans of real cast aluminum parts. Furthermore, with our machine learning methods outperforming the filter-based method on

simulated *and* on real CT scans, we argue that our realistically simulated training data is all we need to train a robust and reliable defect detector.

Putting The Deep Learning Method Through Its Paces. Finally, we provided an extensive discussion about the possibilities which the combination of simulation pipeline and deep learning offers and where its current limitations lie. We examined the eligibility of this approach for in-line scenarios by adapting our model to cope with both: an increased noise-level (i. e. a reduced contrast resolution) and larger focal spot sizes (i. e. a reduced spatial resolution). We saw that with the help of deep learning we can tremendously reduce the necessary scan time and, thus, increase the inspection throughput, which again enables a full in-line inspection of the entire production. The full inspection of the production is beneficial not only for cast aluminum parts. Thus, we further examined what it takes to expand our method to inspect other materials and types of defects. While for some tasks like the inspection of parts produced by plastic injection molding or iron casting our method works out of the box, other tasks like the inspection of additively manufactured parts or the examination of fatigue requires some adaptations of the pipeline, for example, the introduction of additional defect types. Especially when it comes to the distinction of different types of defects, it is necessary to adapt the pipeline. An automated determination whether our deep learning method is able to solve a task on a given CT scan of any given part at any given scan quality by looking at uncertainty scores was not possible. The operating domain expert has to take care of what is being examined and has to ensure that the data quality is sufficient, e. g. by scanning a test specimen on a regular basis.

As the aim of this work is to create a deep learning model for actual productive use and not just beating another high-score in yet another leader board, we will describe the procedure to enable a deep learning based inspection for in-line scenarios in the following section.

7.1. From Theory to Production

Offering a reliable and universal deep learning method for the detection and segmentation of defects in CT scans of cast aluminum parts is not yet viable; too broad is the spectrum of possible scan configurations and the associated appearance of the CT scans. However, on our journey to a more general product we can tailor our method to more specific needs—and we can automate this process. Therefore, we propose the following extended simulation pipeline to raise our method to a production level. Figure 7.1 shows an outline of the extended pipeline. Of course, this approach is tailored towards in-line scenarios, where an increased setup effort is worthwhile, and inspection tasks which cannot be solved with CT otherwise.

To get this pipeline running, fine-tuning our deep learning method for a specific task, the client has to provide us with the following four pieces of information:

- First of all, we need a computer-aided design (CAD) of the part under examination. This can either be the CAD which is used for the casting process or the CAD of the final, machined part. As we do not build a reference-based method, the CAD only to some extent has to correspond to the actual part under examination. For example, we built our replica of the wax part (refer to Section 6.2.2) to just match the dimensions instead of doing a full reverse engineering.

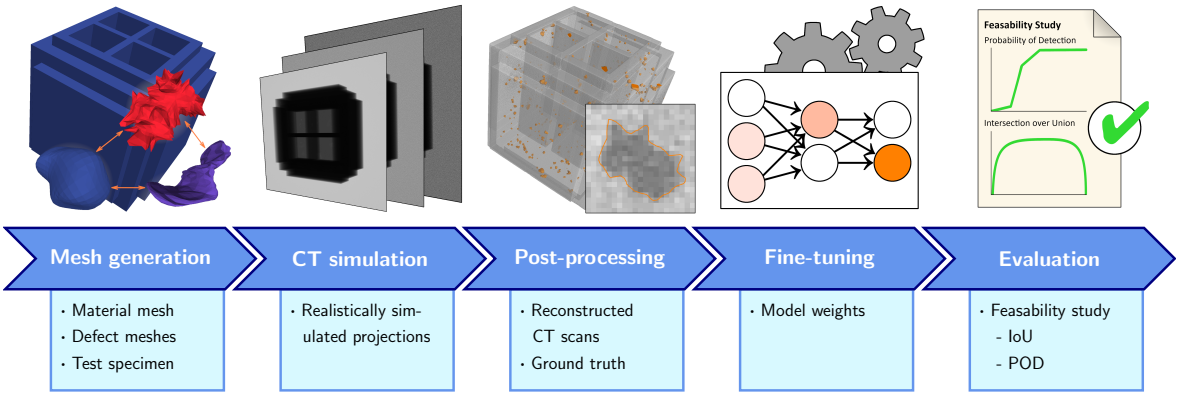


Figure 7.1.: The extended simulation pipeline for production. Besides the already presented steps of mesh generation, CT simulation, and reconstruction and ground truth generation, this extended version further automatically starts fine-tuning the pre-trained model and evaluates the fine-tuned model on task-specific validation data.

Therefore, we still are invariant to minor changes which are done by the casting engineers on a regular basis.

- Furthermore, we need some information of the material of the part under examination for the simulation process. Again, we do not need a precise recipe of the material revealing every top secret ingredient but only a rough composition containing the portions of the material which contribute most to the attenuation.
- Then, we need some information about the expected defect types and defect properties. Are we dealing with casting defects including shrinkage cavities and structural loosening or is it an additively manufactured part mostly containing tiny gas pores? A coarse specification of the size and frequency distributions is helpful as well but not required.
- Finally, we need to know what the CT system which will be used for the inspection task is capable of, or better what CT setup will be used for the actual inspection so that we can tune our simulations to produce CT scans as close to reality as possible.

With these inputs at hand we can start the extended simulation pipeline which not only simulates the CT scans according to the input provided by the client but further fine-tunes and evaluates the model. Therefore, it comprises the following five steps:

1. In the first step of the new production pipeline, instead of procedurally generating virtual die casts, we create a few virtual replicas of the part provided by our client. For each replica we use a different random initialization for the procedural modeling of the defects and their placement. We split these parts into a training set for fine-tuning and a validation set for the IoU evaluation. Based on our experience, we need only very few samples for fine-tuning. Thus, we can keep the simulation cost low and only simulate about five to ten CT scans. In addition, we automatically generate the meshes of a test specimen in accordance with the rules we defined in Section 5.2.2. This will be used for the POD evaluation.
2. In the second step we simulate the replicas of the part as well as the test specimen using slight variations in exposure time, voltage, and other parameters but staying close to our client's setup. The variations are necessary to make the fine-tuned model robust against minor

changes in the data quality of the actual CT scans. Simulating each model multiple times, we get about ten realistically simulated CT scans for training and use the rest for validation. As discussed in Section 6.2.2 this small amount of additional data is absolutely sufficient to create a model that yields reliable predictions. Additionally, we get a simulation of the test specimen with the data quality which we expect to have at our client's site.

3. In the post-processing step, we reconstruct the CT scans and compute the accurate per-voxel ground truth as we did in the original simulation pipeline presented in Section 3.3. Here, we do not need to make any adjustments.

4. After finishing the reconstruction of the training data we can start fine-tuning our model. As basis we use our model which is trained to segment defects in CT scans of cast aluminum parts (see Section 4.3.3). During fine-tuning our model needs about 25 000 iterations at a reduced learning rate of 10^{-5} to converge (compare with Section 6.1.2).

5. Finally, we evaluate the fine-tuned model and create a report which comprises the IoU (refer to Section 5.3) computed on the synthetic validation set as well as the POD (refer to Section 5.2) computed on the simulation of the test specimen. This provides the client with the information about the results that can be expected from the model on the specific inspection task. Optionally, we can do a qualitative evaluation on a real CT scan which has to be provided by our client. At this point we like to note that even though we tailor our model towards a specific task we still have a reference-free approach and, therefore, are not sensible to minor changes in the object under examination, for example, a different placement of the casting gates.

With this pipeline it takes about two days to adapt our deep learning method to a given task, yielding the following artifacts:

- Probably the most important output is the fine-tuned, task-specific defect detection model which can be shipped to the client.

- Additionally, we directly evaluate this model in terms of its POD and its IoU as part of a feasibility study. The evaluations serve for the final acceptance by the client.

- As a side product we further get the realistically simulated CT scans which can be examined and the distilled information can be passed on to improve our more general defect detection model which we use as basis for fine-tuning.

For the final roll-out, we integrate the deep learning method as a plug-in in VGSTUDIO MAX and with that enable it in our client's existing work flow. To get the maximum performance with minimal overhead we switch from TensorFlow (Google Brain), which we use for the training of our models, to TensorRT (NVIDIA). TensorRT allows us to speed up inference, for instance, by using a half-precision floating-point quantization to perform twice as many operations at once or by combining operations in a single step without having to write the intermediate results to the memory, e. g. packing the convolution, the addition of the bias, and the application of the activation function. Furthermore, the prediction engine created by TensorRT is highly optimized for the client's specific accelerator (GPU). We end up with a fast, reliable, and parameter-free, yet still task-specific in-line inspection method which, however, is invariant to changes of details like the casting gates.

7.2. Future Work

In this work we laid the foundation for a far-reaching revolution in NDT and industry 4.0 by showing that realistically simulated CT scans can serve as training data for NDT tasks. There is a boundless list of NDT-tasks in which our combination of simulation pipeline and deep learning can be used to enable a reliable inspection: Most obvious is the transfer to the detection of defects in other domains, e. g. in additive manufacturing or fiber composite materials. Moreover, we are not restricted to modeling defects: For example, we can model short glass or carbon fibers to create more training data for the instance segmentation of individual fibers in a fiber composite material, or we can model more complex parts like specific screws and try to find them in large assembled components or count their occurrences in piles of loose parts. For medical applications we could think of a framework that makes use of simulated CT scans and deep learning to lower the dose for the real patient as described at the end of Section 3.3.3.

Simulation Pipeline. Even though the simulations created by our pipeline already make up good training data, there is still some room to improve the realism of the scans: The refinement of the different types of defects to become more distinct, the adaptation of the mesh generation to other manufacturing techniques like AM, or—as indicated by the predicted confidence (in Section 6.4)—the modeling of mounts. Furthermore, we could take process simulations into account to identify possible porosity hot spots and so improve our graph-drawing approximation. To take the CT simulation to the next level, we can improve the simulation of some artifacts, for instance, by introducing focal drift, imprecise angle steps, or in-detector scattering. Moreover, we would benefit from implementing, for example, helical scan patterns for the simulation of larger parts. However, most importantly, to enable a quick expansion to different fields of CT-based NDT, the simulation pipeline has to be made more easily usable.

Deep Learning Method. Probably the most substantial improvement on the model side would be the reliable detection of even smaller defects, i. e. defects with a radius of less than four voxels. One possible way would be to predict a super-resolution segmentation mask. The essentials for these experiments are already built-in: We just need to create a sub-voxel precise ground truth mask by over-sampling the defect meshes creating 8, 27, or even 64 ground truth voxels per voxel of the CT scan. The available space on the GPU memory is the limiting factor here. In return, we probably are able to further increase the precision of our prediction and maybe can also improve the distinction of different types of defects.

Bibliography

[1] Gianni Nicolettoa, Giancarlo Anzelottia, and Radomila Konečnáb. X-ray Computed Tomography vs. Metallography for Pore Sizing and Fatigue of Cast Al-Alloys. *Procedia Engineering*, 2(1):547–554, 2010.

[2] Yakub Tijani, André Heinrietz, Wolfram Stets, and Patrick Voigt. Detection and Influence of Shrinkage Pores and Nonmetallic Inclusions on Fatigue Life of Cast Aluminum Alloys. *Metallurgical and Materials Transactions A*, 44:5408–5415, 2013.

[3] Doru Michael Stefanescu. Computer Simulation of Shrinkage Related Defects in Metal Castings – A Review. *International Journal of Cast Metals Research*, 18(3):129–143, 2005.

[4] I. Boromei, L. Ceschini, Alessandro Morri, An. Morri, Gianni Nicolettoa, and E. Riva. Influence of the Solidification Microstructure and Porosity on the Fatigue Strength of Al-Si-Mg Casting Alloys. *Metallurgical Science and Technology*, 28(2), 2010.

[5] George F. Vander Voort. *Metallography: Principles and Practice.* ASM International, New York, 1999.

[6] Leonardo DeChiffre, Simone Carmignato, Jean-Pierre Kruth, Robert H. Schmitt, and Albert Weckenmann. Industrial Applications of Computed Tomography. *CIRP Annals – Manufacturing Technology*, 63(2):655–677, 2014.

[7] Johannes Ludwig Vrana. NDE 4.0: Digital Twin, Semantics, Interfaces, Networking, Feedback, New Markets and Integration into the Industrial Internet of Things. *Computing Research Repository (CoRR)*, abs/2004.05193, 2020.

[8] Xinjin Cao and John Campbell. Oxide Inclusion Defects in Al-Si-Mg Cast Alloys. *Canadian Metallurgical Quarterly*, 44:435–448, 2005.

[9] BDG Richtlinie P203. Porositätsanalyse und -beurteilung mittels industrieller Röntgen-Computertomographie (CT). 2019.

[10] D. G. Eskin, L. Katgerman, Suyitno, and J. F. Mooney. Contraction of Aluminum Alloys During and After Solidification. *Metallurgical and Materials Transactions A*, 35:1325–1335, 2004.

[11] Elena Fiorese, Franco Bonollo, Giulio Timelli, Lars Arnberg, and Elisabetta Gariboldi. New Classification of Defects and Imperfections for Aluminum Alloy Castings. *International Journal of Metalcasting*, 9:55–66, 2015.

[12] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3–440, 2015.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based Convolutional Networks for Accurate Object Detection and Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016.

[15] Michael Kampffmeyer, Arnt-Børre Salberg, and Robert Jenssen. Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 680–688, 2016.

[16] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9404–9413, 2019.

[17] Marius Cordts, Mohamed Omran Sebastian Ramos, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.

[19] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *International Conference on 3D Vision (3DV)*, pages 565–571, 2016.

[20] Qi Dou, Hao Chen, Yueming Jin, Lequan Yu, Jing Qin, and Pheng-Ann Heng. 3D Deeply Supervised Network for Automatic Liver Segmentation from CT Volumes. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 149–157, 2016.

[21] Jakob Wasserthal, Peter Neher, and Klaus H. Maier-Hein. TractSeg – Fast and Accurate White Matter Tract Segmentation. *NeuroImage*, 183:239–253, 2018.

[22] Sérgio Pereira, Adriano Pinto, Victor Alves, and Carlos A. Silva. Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251, 2016.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[24] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H. Sebastian Seung. Superhuman Accuracy on the SNEMI3D Connectomics Challenge. *Computing Research Repository (CoRR)*, abs/1706.00120, 2017.

[25] Iasonas Kokkinos. Pushing the Boundaries of Boundary Detection Using Deep Learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[26] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *ACM Conference on Recommender Systems*, pages 101–109, 2019.

[27] Alon Halevy, Peter Norvig, and Fernando Pereira. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24:8–12, 2009.

[28] David Acuna, Amlan Kar, and Sanja Fidler. Devil is in the Edges: Learning Semantic Boundaries From Noisy Annotations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11075–11083, 2019.

[29] Simon A. A. Kohl, Bernardino Romera-Paredes, Klaus H. Maier-Hein, Danilo Jimenez Rezende, S. M. Ali Eslami, Pushmeet Kohli, Andrew Zisserman, and Olaf Ronneberger. A Hierarchical Probabilistic U-Net for Modeling Multi-Scale Ambiguities. *Computing Research Repository (CoRR)*, abs/1902.09063, 2019.

[30] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Martina Marek, and Martin Bokeloh. An Annotation Saved is an Annotation Earned: Using Fully Synthetic Training for Object Detection. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 2787–2796, 2019.

[31] Patrick Fuchs, Thorben Kröger, Tobias Dierig, and Christoph S. Garbe. Generating Meaningful Synthetic Ground Truth for Pore Detection in Cast Aluminum Parts. In *Conference on Industrial Computed Tomography (iCT)*, 2019.

[32] Patrick Fuchs, Thorben Kröger, and Christoph S. Garbe. Self-supervised Learning for Pore Detection in CT-Scans of Cast Aluminum Parts. In *International Symposium on Digital Industrial Radiology and Computed Tomography (DIR)*, 2019.

[33] Thorsten Buzug. *Computed Tomography: From Photon Statistics to Modern Cone-Beam CT*. Springer, Heidelberg, 2008.

[34] Paul Suetens. *Fundamentals of Medical Imaging*. Cambridge University Press, Cambridge, 2009.

[35] Simone Carmignato, Wim Dewulf, and Richard Leach, editors. *Industrial X-Ray Computed Tomography*. Springer International Publishing, Heidelberg, 2018.

[36] EN AW-2014. Aluminium Material Data Sheet EN AW-2014, EN AW-Al Cu4SiMg. 2011.

[37] DGZfP Merkblatt D 06. Anforderungen und Rahmenbedingungen für den Einsatz der Röntgencomputertomographie in der Industrie. 2019.

[38] Shuai Leng, Michael Bruesewitz, Shengzhen Tao, Kishore Rajendran, Ahmed F. Halaweish, Norbert G Campeau, Joel G. Fletcher, and Cynthia H. McCollough. Photon-counting Detector CT: System Design

and Clinical Applications of an Emerging Technology. *RadioGraphics*, 39(3):729–743, 2019.

[39] Katsuyuki Taguchi and Jan S. Iwanczyk. Vision 20/20: Single Photon Counting X-ray Detectors in Medical Imaging. *Medical Physics*, 40(10), 2013.

[40] Pierre Gravel, Gilles Beaudoin, and Jacques A. De Guise. A Method for Modeling Noise in Medical Images. *IEEE Transactions on Medical Imaging*, 23(10):1221–1232, 2004.

[41] Alexander Bub, Sven Gondrom, Michael Maisl, N. Uhlmann, and Walter Arnold. Image Blur in a Flat-panel Detector Due to Compton Scattering at its Internal Mountings. *Measurement Science and Technology*, 18(5):1270–1277, 2007.

[42] Sven Gondrom and Michael Maisl. 3D Reconstructions of Micro-Systems Using X-ray Tomographic Methods. In *World Conference on Non-Destructive Testing (WCNDT)*, 2004.

[43] Sven Gondrom. Physics and Special Technology of Very Fast or Even Inline Industrial 3D-CT. In *Asia Pacific Conference for Non-Destructive Testing (APCNDT)*, 2017.

[44] Michael Reiter, Christian Gusenbauer, Reinhold Huemer, and Johannes Kastner. At-line X-ray Computed Tomography of Serial Parts Optimized by Numerical Simulations. In *International Symposium on Digital Industrial Radiology and Computed Tomography (DIR)*, 2019.

[45] Istvan Szabo, Jiangtao Sun, Guojin Feng, Jamil Kanfoud, Tat-Hean Gan, and Cem Selcuk. Automated Defect Recognition as a Critical Element of a Three Dimensional X-Ray Computed Tomography Imaging-Based Smart Non-Destructive Testing Technique in Additive Manufacturing of Near Net-Shape Parts. *Applied Sciences*, 7(11), 2017.

[46] Tomasz Konopczyński, Thorben Kröger, Lei Zheng, and Jürgen Hesser. Instance Segmentation of Fibers from Low Resolution CT Scans via 3D Deep Embedding Learning. In *British Machine Vision Conference (BMVC)*, page 268, 2018.

[47] Wenhui Hou, Ye Wei, Jie Guo, Yi Jin, and Chang'an Zhu. Automatic Detection of Welding Defects using Deep Neural Network. *Journal of Physics: Conference Series*, 933, 2018.

[48] Thomas Schromm, Fabian Diewald, and Christian Grosse. An Attempt to Detect Anomalies in Car Body Parts Using Machine Learning Algorithms. In *Conference on Industrial Computed Tomography (iCT)*, 2019.

[49] Fei Zhao, Paulo Mendonca, Jie Yu, and Robert Kaucic. Learning-based Automatic Defect Recognition With Computed Tomographic Imaging. In *IEEE International Conference on Image Processing*, pages 2762–2766, 2013.

[50] Hu Chen, Yi Zhang, Mannudeep K. Kalra, Feng Lin, Yang Chen, Peixi Liao, Jiliu Zhou, and Ge Wang. Low-Dose CT With a Residual Encoder-Decoder Convolutional Neural Network. *IEEE Transactions on Medical Imaging*, 36(12):2524–2535, 2017.

[51] Maryam Gholizadeh-Ansari, Javad Alirezaie, and Paul Babyn. Deep Learning for Low-Dose CT Denoising Using Perceptual Loss and Edge Detection Layer. *Journal of Digital Imaging*, 33:504–515, 2020.

[52] Joscha Maier, Stefan Sawall, Michael Knaup, and Marc Kachelrieß. Deep Scatter Estimation (DSE): Accurate Real-Time Scatter Estimation for X-Ray CT Using a Deep Convolutional Neural Network. *Journal of Nondestructive Evaluation*, 37(57), 2018.

[53] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[54] Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Graham Knott, and Pascal Fua. Supervoxel-based Segmentation of Mitochondria in EM Image Stacks with Learned Shape Features. *IEEE Transactions on Medical Imaging*, 31(2):474–486, 2012.

[55] Marius Leordeanu, Rahul Sukthankar, and Christian Sminchisescu. Efficient Closed-form Solution to Generalized Boundary Detection. In *European Conference on Computer Vision (ECCV)*, pages 516–529, 2012.

[56] Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Reviews*, pages 386–408, 1958.

[57] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012.

[59] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[60] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[61] Mark Everingham, Ali S. M. Eslami, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, 2015.

[62] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013.

[63] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. NVIDIA Tesla: A Unified Graphics and Computing Architecture. *IEEE Micro*, 28(2):39–55, 2008.

[64] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 2006.

[65] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[66] Andrew L. Maas, Awni Y. Hannun, and Andrew N. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *International Conference on Machine Learning (ICML)*, 2013.

[67] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[68] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. In *International Conference on Learning Representations (ICLR)*, 2015.

[69] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[70] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[71] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, 2016.

[72] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[73] Veit Sandfort, Ke Yan, Perry J. Pickhardt, and Ronald M. Summers. Data Augmentation Using Generative Adversarial Networks (CycleGAN) to Improve Generalizability in CT Segmentation Tasks. *Scientific Reports*, 9(16884), 2019.

[74] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. volume 39, pages 1137–1149, 2017.

[75] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[76] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[77] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[78] Kevin S. Zhou, Hayit Greenspan, and Dinggang Shen. *Deep Learning for Medical Image Analysis*. Academic Press, Cambridge, 2017.

[79] Björn H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin

Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, Levente Lanczi, Elizabeth Gerstner, Marc-André Weber, Tal Arbel, Brian B. Avants, Nicholas Ayache, Patricia Buendia, D. Louis Collins, Nicolas Cordier, Jason J. Corso, Antonio Criminisi, Tilak Das, Hervé Delingette, Çağatay Demiralp, Christopher R. Durst, Michel Dojat, Senan Doyle, Joana Festa, Florence Forbes, Ezequiel Geremia, Ben Glocker, Polina Golland, Xiaotao Guo, Andac Hamamci, Khan M. Iftekharuddin, Raj Jena, Nigel M. John, Ender Konukoglu, Danial Lashkari, José Antonió Mariz, Raphael Meier, Sérgio Pereira, Doina Precup, Stephen J. Price, Tammy Riklin Raviv, Syed M. S. Reza, Michael Ryan, Duygu Sarikaya, Lawrence Schwartz, Hoo-Chang Shin, Jamie Shotton, Carlos A. Silva, Nuno Sousa, Nagesh K. Subbanna, Gabor Szekely, Thomas J. Taylor, Owen M. Thomas, Nicholas J. Tustison, Gozde Unal, Flor Vasseur, Max Wintermark, Dong Hye Ye, Liang Zhao, Binsheng Zhao, Darko Zikic, Marcel Prastawa, Mauricio Reyes, and Koen Van Leemput. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015.

[80] Patrick Bilic, Patrick Ferdinand Christ, Eugene Vorontsov, Grzegorz Chlebus, Hao Chen, Qi Dou, Chi-Wing Fu, Xiao Han, Pheng-Ann Heng, Jürgen Hesser, Samuel Kadoury, Tomasz Konopczynski, Miao Le, Chunming Li, Xiaomeng Li, Jana Lipkovà, John Lowengrub, Hans Meine, Jan Hendrik Moltz, Chris Pal, Marie Piraud, Xiaojuan Qi, Jin Qi, Markus Rempfler, Karsten Roth, Andrea Schenk, Anjany Sekuboyina, Eugene Vorontsov, Ping Zhou, Christian Hülsemeyer, Marcel Beetz, Florian Ettlinger, Felix Grün, Georgios Kaissis, Fabian Lohöfer, Rickmer Braren, Julian Holch, Felix Hofmann, Wieland Sommer, Volker Heinemann, Colin Jacobs, Gabriel Efrain Humpire Mamani, Bram van Ginneken, Gabriel Chartrand, An Tang, Michal Drozdzal, Avi Ben-Cohen, Eyal Klang, Marianne M. Amitai, Eli Konen, Hayit Greenspan, Johan Moreau, Alexandre Hostettler, Luc Soler, Refael Vivanti, Adi Szeskin, Naama Lev-Cohain, Jacob Sosna, Leo Joskowicz, and Björn H. Menze. The Liver Tumor Segmentation Benchmark (LiTS). *Computing Research Repository (CoRR)*, abs/1901.04056, 2019.

[81] Samuel G. 3rd Armato, Geoffrey McLennan, Luc Bidaut, , Michael F. McNitt-Gray, Charles R. Meyer, Anthony P. Reeves, Binsheng Zhao, Denise R. Aberle, Claudia I. Henschke, Eric A. Hoffman, Ella A. Kazerooni, Heber MacMahon, Edwin J. R. Van Beeke, David Yankelevitz, Alberto M. Biancardi, Peyton H. Bland, Matthew S. Brown, Roger M. Engelmann, Gary E. Laderach, Daniel Max, Richard C. Pais, David P. Y. Qing, Rachael Y. Roberts, Amanda R. Smith, Adam Starkey, Poonam Batrah, Philip Caligiuri, Ali Farooqi, Gregory W. Gladish, C Matilda Jude, Reginald F. Munden, Iva Petkovska, Leslie E. Quint, Lawrence H. Schwartz, Baskaran Sundaram, Lori E. Dodd, Charles Fenimore, David Gur, Nicholas Petrick, John Freymann, Justin Kirby, Brian Hughes, Alessi Vande Casteele, Sangeeta Gupte, Maha Sallamm, Michael D. Heath, Michael H. Kuhn, Ekta Dharaiya, Richard Burns, David S. Fryd, Marcos Salganicoff, Vikram Anand, Uri Shreter, Stephen Vastagh, and Barbara Y. Croft. The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A Completed Reference Database of Lung Nodules on CT Scans. *Medical Physics*, 38(2):915–931, 2011.

[82] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, Heidelberg, 2010.

[83] David G. Lowe. Distinctive Image Features from Scale-invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[84] M. A. Fischler and R. A. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973.

[85] Dariu M. Gavrila and Larry S. Davis. Fast Correlation Matching in Large (Edge) Image Databases. In *AIPR Workshop: Image and Information Systems: Applications and Opportunities*, 1995.

[86] Ondrej Chum and Andrew Zisserman. An Exemplar Model for Learning Object Classes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[87] Bernd Jähne. *Digital Image Processing*. Springer, Heidelberg, 2005.

[88] I. Fogel and D. Sagi. Gabor Filters as Texture Discriminator. *Biological Cybernetics*, 61:103–113, 1989.

[89] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

[90] David G. Lowe. Object Recognition from Local Scale-Invariant Features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[91] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In *European*

Conference on Computer Vision (ECCV), pages 404–417, 2006.

[92] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2009.

[93] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[94] Stuart Berg, Dominik Kutra, Thorben Kröger, Christoph N. Strähle, Bernhard X. Kausler, Carsten Haubold, Martin Schiegg, Janez Ales, Thorsten Beier, Markus Rudy, Kemal Eren, Jaime I. Cervantes, Buote Xu, Fynn Beuttenmueller, Adrian Wolny, Chong Zhang, Ullrich Koethe, Fred A. Hamprecht, and Anna Kreshuk. ilastik: Interactive Machine Learning for (Bio)image Analyis. *Nature Methods*, 16:1226–1232, 2019.

[95] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. Learning to Segment Every Thing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4233–4241, 2018.

[96] Miguel Bautista, Patrick Fuchs, and Björn Ommer. Learning Where to Drive by Watching Others. In *German Conference Pattern Recognition (GCPR)*, pages 29–40, 2017.

[97] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[98] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *European Conference on Computer Vision (ECCV)*, pages 69–84, 2016.

[99] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large-Scale Hierarchical Image Database. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[100] Julien Fauqueur, Gabriel J. Brostow, and Roberto Cipolla. Assisted Video Object Labeling By Joint Tracking of Regions and Keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–7, 2007.

[101] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and Recognition Using Structure from Motion Point Clouds. In *European Conference on Computer Vision (ECCV)*, pages 44–57, 2008.

[102] Tobias Heimann, Bram van Ginneken, Martin A. Styner, Yulia Arzhaeva, Volker Aurich, Christian Bauer, Andreas Beck, Christoph Becker, Reinhard Beichel, György Bekes, Fernando Bello, Gerd Binnig, Horst Bischof, Alexander Bornik, Peter M. M. Cashman, Ying Chi, Andrés Cordova, Benoit M. Dawant, Márta Fidrich, Jacob D. Furst, Daisuke Furukawa, Lars Grenacher, Joachim Hornegger, Dagmar Kainmüller, Richard I. Kitney, Hidefumi Kobatake, Hans Lamecker, Thomas Lange, Jeongjin Lee, Brian Lennon, Rui Li, Senhu Li, Hans-Peter Meinzer, Gábor Nemeth, Daniela S. Raicu, Anne-Mareike Rau, Eva M. van Rikxoort, Mikaël Rousson, László Rusko, Kinda A. Saddi, Günter Schmidt, Dieter Seghers, Akinobu Shimizu, Pieter Slagmolen, Erich Sorantin, Grzegorz Soza, Ruchaneewan Susomboon, Jonathan M. Waite, Andreas Wimmer, and Ivo Wolf. Comparison and Evaluation of Methods for Liver Segmentation from CT datasets. *IEEE Transactions on Medical Imaging*, 28(8):1251–1265, 2009.

[103] Domingo Mery, Vladimir Riffo, Uwe Zscherpel, German Mondragón, Iván Lillo, Irene Zuccar, Hans Lobel, and Miguel Carrasco. GDXray: The Database of X-ray Images for Nondestructive Testing. *Journal of Nondestructive Evaluation*, 34(42), 2015.

[104] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349, 2016.

[105] Stephan Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. In *European Conference on Computer Vision (ECCV)*, pages 102–118, 2016.

[106] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1297–1304, 2011.

[107] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from Simulated and Unsupervised Images Through Adversarial Training. In *IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 2242–2251, 2017.

[108] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[109] Aayush Tremblay, Jonathan Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1082–10828, 2018.

[110] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep Drone Racing: From Simulation to Reality With Domain Randomization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1082–10828, 2018.

[111] Xinyi Ren, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Abhishek Gupta, Aviv Tamar, and Pieter Abbeel. Domain Randomization for Active Pose Estimation. In *International Conference on Robotics and Automation (ICRA)*, pages 7228–7234, 2019.

[112] Gábor J. Székely and Maria L. Rizzoc. Energy Statistics: A Class of Statistics Based on Distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272, 2013.

[113] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised Visual Representation Learning by Context Prediction. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430, 2015.

[114] Aiham Taleb, Winfried Loetzsch, Noel Danz, Julius Severin, Thomas Gärtner, Benjamin Bergner, and Christoph Lippert. 3D Self-Supervised Methods for Medical Imaging. *Computing Research Repository (CoRR)*, abs/2006.03829, 2020.

[115] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable Are Features in Deep Neural Networks? In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 3320–3328, 2014.

[116] Pierre Baldi. Autoencoders, Unsupervised Learning, and DeepArchitectures. In *International Conference on Unsupervised and Transfer Learning Workshop*, pages 37–50, 2011.

[117] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoder. In *International Conference on Machine Learning (ICML)*, pages 1096–1103, 2008.

[118] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016.

[119] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative Image Inpainting with Contextual Attention. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5505–5514, 2018.

[120] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image Inpainting for Irregular Holes Using Partial Convolutions. In *European Conference on Computer Vision (ECCV)*, pages 89–105, 2018.

[121] Jaehoon Choi, Taekyung Kim, and Changick Kim. Self-Ensembling With GAN-Based Data Augmentation for Domain Adaptation in Semantic Segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 6829–6839, 2019.

[122] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *International Conference on Machine Learning (ICML)*, pages 1989–1998, 2018.

[123] Benjamin Planche, Ziyan Wu, Kai Ma, Shanhui Sun, Stefan Kluckner, Terrence Chen, Andreas Hutter, Sergey Zakharov, Harald Kosch, and Jan Ernst. DepthSynth: Real-Time Realistic Synthetic Data Generation from CAD Models for 2.5D Recognition. In *International Conference on 3D Vision (3DV)*, 2017.

[124] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real Single-Image Flight Without a Single Real Image. In *Robotics: Science and Systems XIII, Massachusetts Institute of Technology*, 2017.

[125] Douglas Heaven. Why Deep-Learning AIs Are So Easy to Fool. *Nature*, 574(7777):163–166, 2019.

[126] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization and Generative Models for Robotic Grasping. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489, 2018.

[127] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014.

[128] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *Proceedings of Machine Learning Research*, pages 214–223, 2017.

[129] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

[130] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[131] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems 29*, pages 2234–2242, 2016.

[132] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. *Computing Research Repository (CoRR)*, abs/1611.02163, 2016.

[133] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode Regularized Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[134] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 5769–5779, 2017.

[135] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing Training of Generative Adversarial Networks through Regularization. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 2018–2028, 2017.

[136] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 3308–3318, 2017.

[137] David Berthelot and Luke Schumm, Thomas Metz. BEGAN: Boundary Equilibrium Generative Adversarial Networks. *Computing Research Repository (CoRR)*, abs/1703.10717, 2017.

[138] Deniz Neufeld, Tobias Würfl1, Roland Gruber, Tobias Schön, and Andreas Maier. Realistic Image Synthesis of Imperfect Specimens using Generative Networks. In *Conference on Industrial Computed Tomography (iCT)*, 2019.

[139] Domingo Mery. *Computer Vision for X-Ray Testing*. Springer International Publishing, Heidelberg, 2015.

[140] Carsten Bellon and Gerd-Rüdiger Jaenisch. aRTist – Analytical RT Inspection Simulation Tool. In *International Symposium on Digital Industrial Radiology and Computed Tomography (DIR)*, 2007.

[141] Carsten Bellon, Andreas Deresch, Christian Gollwitzer, and Gerd-Rüdiger Jaenisch. Radiographic Simulator aRTist: Version 2. In *World Conference on Nondestructive Testing (WCNDT)*, 2012.

[142] Stefan Kasperl, Richard Schielein, Frank Sukowski, Peter Hornberger, and Andreas Gruber. CT Simulation Study to Demonstrate Material Impact Using Hole Plates. In *European Conference on Nondestructive Testing (ECNDT)*, 2014.

[143] Tobias Schönfeld and Markus Bartscher. Verification and Application of Quality Measures in Dimensional Computed Tomography. In *International Symposium on Digital Industrial Radiology and Computed Tomography (DIR)*, 2015.

[144] Matthias Fleßner, Andreas Müller, Daniela Götz, Eric Helmecke, and Tino Hausotte. Assessment of the Single Point Uncertainty of Dimensional CT Measurements. In *Conference on Industrial Computed*

Tomography (iCT), 2016.

[145] Jason Weber and Joseph Penn. Creation and Rendering of Realistic Trees. In *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 119–128, 1995.

[146] Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. A Survey on Procedural Modeling for Virtual Worlds. *Computer Graphics Forum*, 33(6):31–50, 2014.

[147] Lars Krecklau and Leif Kobbelt. Procedural Modeling of Interconnected Structures. *Computer Graphics Forum*, 30(2):335–344, 2011.

[148] Hichem Barki, Gael Guennebaud, and Sebti Foufou. Exact, Robust, and Efficient Regularized Booleans On General 3D Meshes. *Computers and Mathematics with Applications*, 70(6):1235–1254, 2015.

[149] ASTM Standarad E505-15. Standard Reference Radiographs for Inspection of Aluminum and Magnesium Die Castings. 2015.

[150] Stefan Guth and Karl-Heinz Lang. An Approach to Lifetime Prediction for a Wrought Ni-base Alloy Under Thermo-mechanical Fatigue With Various Phase Angles Between Temperature and Mechanical Strain. *International Journal of Fatigue*, 99(2):286–294, 2016.

[151] Thomas M. J. Fruchterman and Edward M. Reingold. Graph Drawing by Force-directed Placement. *Software—Practice and Experience*, 21(1):1129–1164, 1991.

[152] M. J. Flynn, S. M. Hames, S. J. Wilderman, and J. J. Ciarelli. Quantum Noise in Digital X-ray Image Detectors With Optically Coupled Scintillators. *IEEE Transactions on Nuclear Science*, 43(4):2320–2325, 1996.

[153] Julia F. Barrett and Nicholas Keat. Artifacts in CT: Recognition and Avoidance. *RadioGraphics*, 24(6):1679–1691, 2004.

[154] F. Edward Boas and Dominik Fleischmann. CT Artifacts: Causes and Reduction Techniques. *Imaging in Medicine*, 4(2):229–240, 2012.

[155] Baerbel Kratz, Frank Herold, Jason C. Robbins, and Jan Tamm. Study on the Influence of Scattered Radiation and the Usage of Scatter Reduction Methods for Computed Tomography. In *Conference on Industrial Computed Tomography (iCT)*, 2017.

[156] Gabriel Probst, Bart Boeckmans, Jean-Pierre Kruth, and Wim Dewulf. Compensation of Drift in an Industrial Computed Tomography System. In *Conference on Industrial Computed Tomography (iCT)*, 2016.

[157] Nadia Flay, Wenjuan Sun, Stephen Brown, Richard Leach, and Thomas Blumensath. Investigation of the Focal Spot Drift in Industrial Cone-beam X-ray Computed Tomography. In *International Symposium on Digital Industrial Radiology and Computed Tomography (DIR)*, 2015.

[158] Karim Zarei Zefreh1, Jan De Beenhouwer, Federica Marone Welford, and Jan Sijbers. Investigation on Effect of Scintillator Thickness on Afterglow in Indirect-Flat Panel Detectors. In *Conference on Industrial Computed Tomography (iCT)*, 2016.

[159] Nobutaka Mitsuhashi, Kaori Fujieda, Takuro Tamura, Shoko Kawamoto, Toshihisa Takagi, and Kousaku Okubo. BodyParts3D: 3D Structure Database for Anatomical Concepts. *Nucleic Acids Research*, 37, 2009.

[160] Ting Luo, Changrong Shi, Xing Zhao, Yunsong Zhao, and Jinqiu Xu. Automatic Synthesis of Panoramic Radiographs From Dental Cone Beam Computed Tomography Data. *Public Library of Science One*, 11(6), 2016.

[161] Yanbo Zhang and Hengyong Yu. Convolutional Neural Network Based Metal Artifact Reduction in X-Ray Computed Tomography. *IEEE Transactions on Medical Imaging*, 37(6):1370–1381, 2018.

[162] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A Non-Local Algorithm for Image Denoising. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 60–65, 2005.

[163] Jerome Darbon, Alexandre Cunha, Tony F. Chan, Stanley J. Osher, and Grant J. Jensen. Fast Nonlocal Filtering Applied to Electron Cryomicroscopy. In *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1331–1334, 2008.

[164] Leonid I. Rudin, Stanley J. Osher, and Emad Fatemi. Nonlinear Total Variation Based Noise Removal Algorithms. *Physica D: Nonlinear Phenomena*, 60(1–4):259–268, 1992.

[165] Luminita A. Vese and Stanley J. Osher. Image Denoising and Decomposition with Total Variation

Minimization and Oscillatory Functions. *Journal of Mathematical Imaging and Vision*, 20(1):7–18, 2004.

[166] Jaakko Sauvola and Matti Pietikäinen. Adaptive Document Image Binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[167] Derek Bradley and Gerhard Roth. Adaptive Thresholding using the Integral Image. *Journal of Graphics Tools*, 12(2):13–21, 7.

[168] Rafael C. Gonzales and Richard E. Woods. *Digital Image Processing*. Prentice Hal, Upper Saddle River, 2007.

[169] Paul Viola and Michael Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[170] Bruno Lay. *Image Processing: A Key to Success in Industrial Applications*, pages 341–352. 1994.

[171] Robert M. Haralick, Stanley R. Sternberg, and Xinhua Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 9(4):532–550, 1987.

[172] Hanno Scharr. *Optimal Operators in Digital Image Processing*. dissertation, Heidelberg University, 2000.

[173] Zhengqin Li and Jiansheng Chen. Superpixel Segmentation using Linear Spectral Clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1356–1363, 2015.

[174] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class Segmentation and Object Localization with Superpixel Neighborhoods. In *IEEE International Conference on Computer Vision (ICCV)*, pages 670–677, 2009.

[175] Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Object Class Segmentation using Random Forests. In *British Machine Vision Conference (BMVC)*, pages 54.1–54.10, 2008.

[176] Nobuyuki Otsu. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[177] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621, 1973.

[178] Ron Goldman. Curvature Formulas for Implicit Curves and Surfaces. *Computer Aided Geometric Design*, 22:632–658, 2005.

[179] Jan J. Koenderink and Andrea J. van Doorn. Surface Shape and Curvature Scales. *Image Vision Computing*, 10(8):557–565, 1992.

[180] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding Variable Importances in Forests of Randomized Trees. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 431–439, 2013.

[181] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. *Computing Research Repository (CoRR)*, abs/1611.10080, 2016.

[182] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[183] Paul F. Jaeger, Simon A. A. Kohl, Sebastian Bickelhaupt, Fabian Isensee, Tristan Anselm Kuder, Heinz-Peter Schlemmer, and Klaus H. Maier-Hein. Retina U-Net: Embarrassingly Simple Exploitation of Segmentation Supervision for Medical Object Detection. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 171–183, 2019.

[184] Nabila Abraham and Naimul Mefraz Khan. A Novel Focal Tversky Loss Function with Improved Attention U-Net for Lesion Segmentation. *Computing Research Repository (CoRR)*, abs/1810.07842, 2018.

[185] Konstantinos Kamnitsas, Wenjia Bai, Enzo Ferrante, Steven G. McDonagh, Matthew Sinclair, Nick Pawlowski, Martin Rajchl, Matthew C. H. Lee, Bernhard Kainz, Daniel Rueckert, and Ben Glocker. Ensembles of Multiple Models and Architectures for Robust Brain Tumour Segmentation. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 450–462, 2017.

[186] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[187] Ricardo Guerrero, Chen Qin, Ozan Oktay, C. Bowles, L. Chen, Richard Joules, Robin Wolz, M. Valdes-

Hernandez, David Dickie, J. Wardlaw, and D. Rueckert. White Matter Hyperintensity and Stroke Lesion Segmentation and Differentiation Using Convolutional Neural Networks. *NeuroImage*, 17:918–934, 2017.

[188] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 562–570, 2015.

[189] Peiyun Hu and Deva Ramanan. Finding Tiny Faces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1522–1530, 2017.

[190] Raghav Mehta and Jayanthi Sivaswamy. M-net: A Convolutional Neural Network for Deep Brain Structure Segmentation. In *International Symposium on Biomedical Imaging (ISBI)*, pages 437–440, 2017.

[191] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

[192] Ryuhei Hamaguchi, Aito Fujita, Keisuke Nemoto, Tomoyuki Imaizumi, and Shuhei Hikosaka. Effective Use of Dilated Convolutions for Segmenting Small Object Instances in Remote Sensing Imagery. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1442–1450, 2018.

[193] Ning Xu, Brian L. Price, Scott Cohen, and Thomas S. Huang. Deep Image Matting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 311–320, 2017.

[194] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard S. Zemel. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 4898–4906, 2016.

[195] Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger. Improving Unsupervised Defect Segmentation by Applying Structural Similarity To Autoencoder. *Computing Research Repository (CoRR)*, abs/1807.02011, 2018.

[196] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.

[197] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[198] Ralph Neuneier and Hans Georg Zimmermann. *How to Train Neural Networks*, pages 373–423. 1998.

[199] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248, 2017.

[200] Ken C. L. Wong, Mehdi Moradi, Hui Tang, and Tanveer Syeda-Mahmood. 3D Segmentation with Exponential Logarithmic Loss for Highly Unbalanced Object Sizes. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 612–619, 2018.

[201] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks. In *Machine Learning in Medical Imaging*, pages 379–387, 2017.

[202] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum Learning. In *International Conference on Machine Learning (ICML)*, pages 41–48, 2009.

[203] Denis Kiefel, M. Scius-Bertrand, and Rainer Stößel. Computed Tomography of Additive Manufactured Components in Aeronautic Industry. In *Conference on industrial CT (iCT)*, 2018.

[204] Denis Kiefel. *Quantitative Porosity Characterization of CFRP Materials using Micro Computed Tomography*. dissertation, Technische Universität München, 2017.

[205] Mark Everingham, Ali S. M. Eslami, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[206] Aaron Carass, Snehashis Roy, Adrian Gherman, Jacob C. Reinhold, Andrew Jesson, Tal Arbel, Oskar Maier, Heinz Handels, Mohsen Ghafoorian, Bram Platel, Ariel Birenbaum, Hayit Greenspan, Dzung L. Pham, Ciprian M. Crainiceanu, Peter A. Calabresi, Jerry L. Prince, William R. Gray Roncal, and Russell T. Shinohara. Evaluating White Matter Lesion Segmentations with Refined Sørensen-Dice Analy-

sis. *Scientific Reports*, 10(8242), 2020.

[207] Jochen H. Kurz, Anne Jüngert, Sandra Dugan, and Gerd Dobmann. Probability of Detection (POD) Determination Using Ultrasound Phased Array for Considering NDT in Probabilistic Damage Assessments. In *World Conference on Nondestructive Testing (WCNDT)*, 2012.

[208] David S. Forsyth and John C. Aldrin. Build Your Own POD. In *European-American Workshop on Reliability of NDE*, 2009.

[209] Alan P. Berens. NDE Reliability Data Analysis. *ASM Handbook: Nondestructive Evaluation and Quality Control*, pages 689–701, 1994.

[210] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *International Conference on Machine Learning*, 2006.

[211] Takaya Saito and Marc Rehmsmeier. The Precision-Recall Plot is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *Public Library of Science One*, 10(3), 2015.

[212] Peter A. Flach and Meelis Kull. Precision-Recall-Gain Curves: PR Analysis Done Right. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 838–846, 2015.

[213] Tom Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[214] F. Timischl. The Contrast-to-Noise Ratio for Image Quality Evaluation in Scanning Electron Microscopy. *Journal of Scanning Microscopy*, 37(1):54–62, 2015.

[215] ASTM Standarad E1441-19. Standard Guide for Computed Tomography (CT). 2019.

[216] ASTM Standarad E1695-95(2013). Standard Test Method for Measurement of Computed Tomography (CT) System Performance. 2013.

[217] H. Steinlechner, G. Haaser, Bernd Oberdorfer, Habe Daniel, S. Maierhofer, M. Schwärzler, and Eduard Gröller. A Novel Approach for Immediate, Interactive CT Data Visualization and Evaluation using GPU-based Segmentation and Visual Analysis. In *Conference on Industrial Computed Tomography (iCT)*, 2019.

[218] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *International Conference on Machine Learning (ICML)*, pages 647–655, 2014.

[219] DIN EN 1561:2012-01. Founding – Grey Cast Irons; German Version EN 1561:2011. 2012.

[220] Robert Snell, Sam Tammas-Williams, Lova Chechik, Alistair Lyle, Everth Hernández-Nava, Charlotte Boig, George Panoutsos, and Iain Todd. Methods for Rapid Pore Classification in Metal Additive Manufacturing. *The Journal of The Minerals, Metals and Materials Society*, 72:101–109, 2020.

[221] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *Computing Research Repository (CoRR)*, abs/1708.02002, 2017.

[222] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016.

[223] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. *Computing Research Repository (CoRR)*, abs/1706.04599, 2017.

[224] Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 5574–5584, 2017.

[225] Armen Der Kiureghiana and Ove Ditlevsen. Aleatory or Epistemic? Does It Matter? *Structural Safety*, 31:105–112, 2009.

[226] Michael Kampffmeyer, Arnt-Børre Salberg, and Robert Jenssen. Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 680–688, 2016.

[227] Tanya Nair, Doina Precup, Douglas L. Arnold, and Tal Arbel. Exploring Uncertainty Measures in Deep Networks for Multiple Sclerosis Lesion Detection and Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 655–663, 2018.

[228] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Conference on Neural Information Processing Systems*

(NeurIPS), pages 6402–6413, 2017.

[229] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric Uncertainty Estimation with Test-time Augmentation for Medical Image Segmentation with Convolutional Neural Networks. *Neurocomputing*, 338:34–45, 2019.

[230] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward Multimodal Image-to-image Translation. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 465–476, 2017.

[231] Terrance DeVries and Graham W. Taylor. Learning Confidence for Out-of-Distribution Detection in Neural Networks. *Computing Research Repository (CoRR)*.

[232] Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

A. Data Atlas

Throughout this thesis we mostly use our main data set comprising 675 realistically simulated CT scans of virtual cast aluminum parts for training and another 54 CT scans for validation. In addition, we have two real CT scans of a pivot cap and an aluminum sample for validation, which were labeled by eight experts (see Table A.1). However, especially in Chapter 6 we utilize a lot of different data sets to explore the vast amount of possibilities which our approach of combining realistic CT simulations with deep learning offers for NDT. Therefore, we provide a comprehensive list of all data sets (CT scans) which are used throughout this work.

Aluminum Cast

The cast aluminum data set represents the core of this work. It is used to explore the principles of training a deep neural network for the detection of casting defects solely on simulated data and comprises simulated training and validation data as well as real CT scans for evaluation, as listed in Table A.1. The original deep learning method was trained on this data.

#	size [vox]	material	domain	type	labels	introduced in
675	1000^3	aluminum	simulated	training	computed	Section 3.3
	The training set contains 418 095 defects, which we use to train our machine-learning-based defect detection in Section 4.2.1 and Section 4.2.2 algorithms, and the "original" deep learning method in Section 4.3.3 in particular.					
54	1000^3	aluminum	simulated	validation	computed	Section 3.3
	The validation set contains another 35 586 defects, which are used in Chapter 5 to evaluate the IoU (Table 5.1, Figure 5.7), the POD (Figure 5.3), the iIoU (Table 5.2, Figure 5.10), and the PR curve (Figure 5.12).					
	Furthermore, we evaluate the influence of different types of artifacts (Section 5.3.2, Figure 5.9) and the performance of different deep learning architectures (Section 5.3.4, Figure 5.11) on this data.					
	Finally, the data is used as example data for aluminum when comparing the influence of different materials (Section 6.2.1, Figure 6.6a). Examples are shown in Figure A.1.					
3	1000^3	aluminum	simulated	test specimen	computed	Section 5.2.2
	The test specimens are used to obtain more consistent results for the POD (Section 5.2.2, Figures 5.5 and 5.6) and to evaluate aluminum data when comparing the influence of different materials (Section 6.2.1, Figure 6.6b).					
2	~ 800^3	aluminum	real	validation	expert annotations	Section 3.1.2
	To see how well our models transfer to the real world, we evaluate them on two real CT scans of a pivot cap and an aluminum sample. In Chapter 5 we compute the IoU (Table 5.1) and the iIoU (Table 5.2) and show some segmentation masks as example (Figure 5.8).					

	Furthermore, we use this data for qualitative evaluation of the defect classification (Section 6.3, Figure 6.16) and uncertainty estimation (Section 6.4, Figures 6.17, 6.18, and 6.19).					
1	~ 1000^3	aluminum	real	validation	metallography	Section 5.5
	This data set comes with a metallographic cut image. We use it to compare our method to a destructive method in terms of the overall porosity (Section 5.5, Figure 5.13).					

Table A.1.: The cast aluminum data set for training and evaluation.

Fast CT Scans

To evaluate how well the different models can deal with a reduction of the scan-time, we evaluated their results in terms of the IoU on two sets of CT scans with decreasing scan-time. An artificial one with a reduced exposure time per projection and a real one with a reduced number of projections, as listed in Table A.2.

#	size [vox]	material	domain	type	labels	introduced in
14	1000^3	aluminum	simulated	validation	computed	Section 6.1.1
	To explore possible scan time reductions we evaluate the IoU as a function of the exposure time, reducing the exposure time per projection in each CT scan (Section 6.1.1, Figure 6.2a).					
8	~ 800^3	aluminum	real	validation	expert annotations	Section 6.1.1
	To explore possible scan time reductions we evaluate the IoU as a function of the number of projections which are used to reconstruct the CT scan (Section 6.1.1, Figure 6.2b).					

Table A.2.: The collection of CT scans of a varying total scan-time.

Low Data Quality

Typical CT scans of in-line scenarios are more blurred and more prone to noise than the data used to train our original deep learning method. Therefore, we introduced a set of especially challenging CT scans to fine-tune the model for in-line scenarios, as listed in Table A.3.

#	size [vox]	material	domain	type	labels	introduced in
50	1000^3	aluminum	simulated	training	computed	Section 6.1.2
	The data set contains particularly artifact-afflicted CT scans with high noise and low spatial resolution to fine-tune the deep learning model for in-line scenarios (Section 6.1.2, Figure 6.4).					
1	~ 600^3	aluminum	real	validation	(qualitative)	Section 6.1.2
	We use this data to qualitatively evaluate the model which was fine-tuned on the training data above on a real in-line CT scan (Section 6.1.2, Figure 6.5).					

Table A.3.: CT scans resembling and taken from in-line scenarios.

Plastic Injection Molding

Beside light metal casting, plastic injection molding is a prominent application of NDT via CT. Therefore, we examine how well our model performs on CT scans of injection molding parts (without regarding different defect types), as listed in Table A.4.

#	size [vox]	material	domain	type	labels	introduced in
54	1000^3	plastic	simulated	validation	computed	Section 6.2.1
	This data set is used as example data for plastics when comparing the influence of different materials in terms of the IoU (Section 6.2.1, Figure 6.6a). Examples are shown in Figure A.1.					
3	1000^3	plastic	simulated	test specimen	computed	Section 6.2.1
	The test specimens are used to evaluate the POD on plastic data when comparing the influence of different materials (Section 6.2.1, Figure 6.6b).					

Table A.4.: Less artifact-afflicted CT scans of plastic injection molded parts.

Iron Cast

The denser the materials, the harder they are to penetrate and the more artifact-afflicted become the CT scans. Therefore, we examine how well our model performs on CT scans of cast iron parts (without regarding different defect types), as listed in Table A.5. Moreover, we see what fine-tuning can do in such a case.

#	size [vox]	material	domain	type	labels	introduced in
50	1000^3	iron	real	training	computed	Section 6.2.1
	The data set contains CT scans which show typical iron cast artifacts like high beam hardening which we use to fine-tune the deep learning model for iron casts (Section 6.2.1).					
54	1000^3	iron	simulated	validation	computed	Section 6.2.1
	This data set is used as example data for iron when comparing the influence of different materials in terms of the IoU (Section 6.2.1, Figure 6.6a) and to validate the fine-tuned models (Section 6.2.1, Figure 6.6c). Examples are shown in Figure A.1.					
3	1000^3	iron	simulated	validation	computed	Section 6.2.1
	The test specimens are used to evaluate the POD on iron data when comparing the influence of different materials (Section 6.2.1, Figure 6.6b) and to compare the POD of the fine-tuned models (Section 6.2.1, Figure 6.6d).					
1	~ 800^3	iron	real	validation	(qualitative)	Section 6.2.1
	We use this data set to provide at least a qualitative evaluation on a real CT scan of a cast iron part (Section 6.2.1, Figure 6.7).					

Table A.5.: Simulated CT scans of cast iron for training and fine-tuning and a real CT scan for a qualitative evaluation.

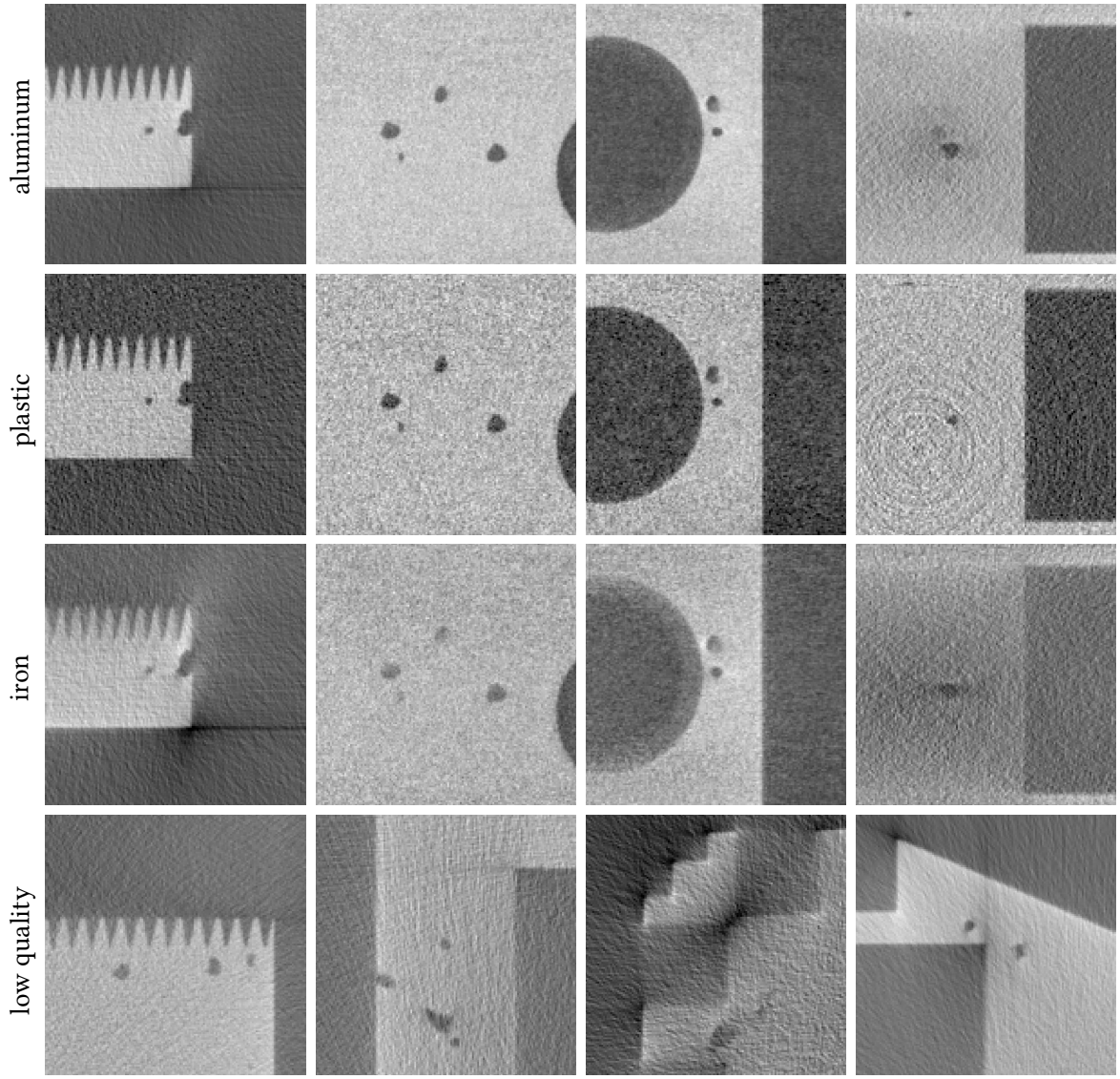


Figure A.1.: The first three rows show slices from the validation set which are simulated with different materials. The plastic data set has the crispest edges and the least cupping artifacts, while the iron data suffers from sever cupping due to strong beam hardening effects. The last row shows some slices from the additional aluminum cast training data of low data quality, which resembles typical in-line scenario CT scans.

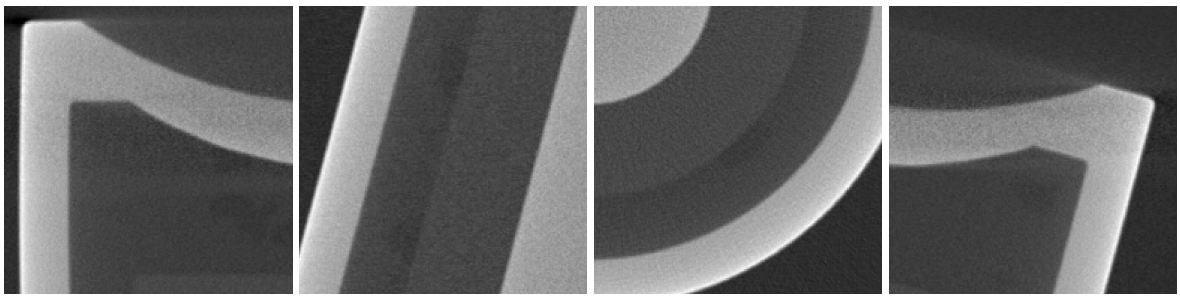


Figure A.2.: The simulated replicas of the multi-material part (a thermistor made of copper, wax, and plastic) show the same strong artifacts in the target region as the original CT scans.

Multi-material Components

With fine-tuning as a well-working method for the adaption to hard cases, we further evaluate the approach on a particularly challenging task: the segmentation of defects in a CT scan of a multi-material component, as listed in Table A.6.

#	size [vox]	material	domain	type	labels	introduced in
3	1000^3	various†	simulated	training	computed	Section 6.2.2
	This data set contains the replicas of a particularly challenging multi-material part, on which we fine-tune our deep learning method (Section 6.2.2, Figure 6.8). Examples are shown in Figure A.2.					
1	~ 900^3	various†	real	validation	(qualitative)	Section 6.2.2
	This is a CT scan of the real multi-material part for which we provide a qualitative evaluation of the fine-tuned model (Section 6.2.2, Figure 6.8).					

Table A.6.: A small set of training data for the evaluation on a particularly hard CT scan of a multi-material component. The low contrast further complicates the detection of the defects.
†: copper, wax, and plastic

Additive Manufacturing

Different manufacturing techniques introduce different kinds of defects. We evaluate how well our deep learning method deals with unknown defect types by evaluating a CT scan of an additively manufactured part, as listed in Table A.7.

#	size [vox]	material	domain	type	labels	introduced in
1	~ 2000^3	titan	real	validation	(qualitative)	Section 6.2.3
	We use this for the qualitative evaluation of additively manufactured data (Section 6.2.3, Figures 6.9 and 6.10).					

Table A.7.: An additively manufactured part with unknown defect types.

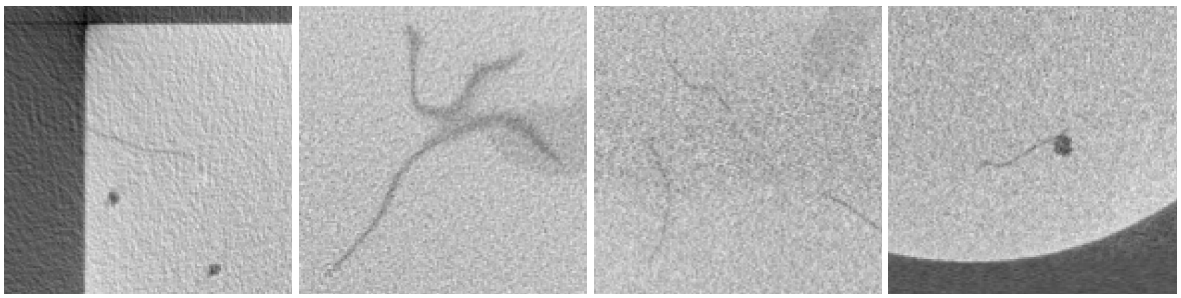


Figure A.3.: Beside gas pores, shrinkage cavities, and solidification cracks these CT scans show large splits which resemble typical fatigue cracks. The data set further comprises different scan qualities.

Fatigue Cracks

Defects do not only occur during the casting but also as a result of material fatigue. The large splits emerging in the material due to fatigue, however, require a special treatment to be detected in the CT scans. Therefore, we introduced a separate data set with large splits, as listed in Table A.8.

#	size [vox]	material	domain	type	labels	introduced in
675	1000^3	aluminum	simulated	training	computed	Section 6.2.4
	This training set additionally contains large splits, which we use to train a model for the detection of fatigue cracks from scratch (Section 6.2.4, Figure 6.11). Examples are shown in Figure A.3.					
10	1000^3	aluminum	simulated	validation	computed	Section 6.2.4
	We use this reduced validation set for a brief quantitative evaluation of the IoU of the detection of cracks (Section 6.2.4).					
5	~ 650^3	nickel	real	validation	(qualitative)	Section 6.2.4
	To provide a qualitative evaluation of the detection of fatigue cracks we examine the CT scans of five tension rods made of a nickel-based alloy (Section 6.2.4, Figure 6.12).					

Table A.8.: Fatigue cracks can emerge in cast metal parts and require a special treatment to be recognized.

B. Model Zoo

In Section 6.1 and Section 6.2 we evaluate our deep learning model at plenty of different states. Originating from the "original" model that was trained for quality laboratory scenario CT scans of cast aluminum parts, we fine-tune it for CT scans of lower data quality (i. e. in-line scenario CT scans), for cast iron parts, and for particularly challenging multi-material parts. Furthermore, we train separate models for the detection of very tiny defects and for thin fatigue cracks. Table B.1 provides an overview of these models, their configuration, and where the corresponding evaluations can be found. All these models use the two-step architecture introduced in Section 4.3.1. See Figure 4.14 for an overview.

name	**trained on**	**fine-tuned on**	**results**[†]
deep learning method ("original")	aluminum cast (Table A.1)	–	Section 6.1.1, Figure 6.3 Section 6.1.2, Figure 6.5b Section 6.2.1, Figures 6.6a and 6.6b Section 6.2.1, Figures 6.6c and 6.6d Section 6.2.1, Figure 6.7a Section 6.2.3, Figures 6.10a and 6.10c
low data quality	aluminum cast (Table A.1)	low data quality (Table A.3)	Section 6.1.1, Figure 6.3, Section 6.1.2, Figure 6.5c Section 6.2.1, Figures 6.6c and 6.6d
iron cast	aluminum cast (Table A.1)	iron cast (Table A.5)	Section 6.2.1, Figures 6.6c and 6.6d Section 6.2.1, Figures 6.7b and 6.7c
thermistor	aluminum cast (Table A.1)	thermistor replica (Table A.6)	Section 6.2.2, Figure 6.8
tiny defects	aluminum cast (Table A.1) (iIoU loss)	–	Section 6.2.3, Figure 6.10b
fatigue crack detection	large splits (Table A.8)	–	Section 6.2.4, Figure 6.12

Table B.1.: The different deep learning models used throughout Section 6.1 and Section 6.2.
[†]: The list is limited to the results shown in Chapter 6.