DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 945

Franz Wagner

Segmentation in Tomography Data: Exploring Data Augmentation for Supervised and Unsupervised Voxel Classification with Neural Networks

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN 978-3-7696-5357-1

DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 945

Segmentation in Tomography Data: Exploring Data Augmentation for Supervised and Unsupervised Voxel Classification with Neural Networks

Dissertation to achieve the academic degree
Doktor-Ingenieur (Dr.-Ing.)
of the Dresden University of Technology (TUD). Faculty of Environmental Sciences

by

Franz Wagner

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN 978-3-7696-5357-1

Adresse des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften:

DGK

Ausschuss Geodäsie (DGK) der Bayerischen Akademie der Wissenschaften
Alfons-Goppel-Straße 11 ● D – 80 539 München
Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100
e-mail post@dgk.badw.de ● http://www.dgk.badw.de

Referee: Prof. Dr. sc. techn. habil. Hans-Gerd Maas, TUD Dresden University of Technology, Institute of Photogrammetry and Remote Sensing

Referee: apl. Prof. Dr. techn. habil. Franz Rottensteiner, Leibniz University Hannover, Institute of Photogrammetry and Geoinformation

Referee: Dr.-Ing. habil. Martin Weinmann, Karlsruhe Institute of Technology, Institute of Photogrammetry and Remote Sensing

Defended on: 12.07.2024

Diese Dissertation ist auf dem Server des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften unter <http://dgk.badw.de/>
sowie auf dem Server der Technischen Universität Dresden unter
<https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-936719> elektronisch publiziert

ISSN 0065-5325 ISBN 978-3-7696-5357-1

Declaration of Conformity

The conformity of this version with the original dissertation on the topic:

"Segmentation in Tomography Data: Exploring Data Augmentation for Supervised and Unsupervised Voxel Classification with Neural Networks"

is hereby confirmed.

Dresden, 30.08.2024

Franz Wagner

Statement of the Authorship

I, Franz Wagner, hereby certify that I have authored this dissertation entitled
"Segmentation in Tomography Data: Exploring Data Augmentation for Supervised and Unsupervised Voxel Classification with Neural Networks"
independently and without additional resources and references than explicitly declared as such in the text by citation or footnote. I have marked both literal and accordingly adopted quotations as such. Unless otherwise stated, all graphics and illustrations were created by myself. The individual author contributions to the selected publications used in this cumulative dissertation are declared in the following section. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Dresden, 30.08.2024

Franz Wagner

Author’s Contribution

This section declares the author’s contribution to the individual scientific articles that form the main part of the thesis. For the sake of convenience the author of the dissertation, Franz Wagner, will be referred to as "the author". The statements follow the "CRediT" guidelines by Brand et al. [1] (2015), Learned Publishing 28(2) (https://doi.org/10.1087/20150211).

Paper 1 [2] (Chapter 4) entitled "River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs" has been published in the *International Journal of Applied Earth Observation and Geoinformation*, an international web-based, open-access, peer-reviewed journal published by Elsevier Science Publishers Amsterdam (NL).
The author is responsible for: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing – original draft, and visualization. All authors contributed to: writing – review & editing. Anette Eltner and Hans-Gerd Maas are responsible for: resources, supervision, and funding acquisition. Hans-Gerd Maas is responsible for: project administration.[1]

Paper 2 [3] (Chapter 5) entitled "A COMPARATIVE STUDY OF DEEP ARCHITECTURES FOR VOXEL SEGMENTATION IN VOLUME IMAGES" has been published in the *ISPRS Archives*, an open-access collection of conference papers published by Copernicus Publications, Göttingen (DE), on behalf of the International Society of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS). The submitted paper was double-blind peer-reviewed. The paper was presented at the ISPRS Geospatial Week 2023 in Cairo, Egypt.
The author is responsible for: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing – original draft, and visualization. All authors contributed to: writing – review & editing. Hans-Gerd Maas is responsible for: resources, supervision, funding acquisition, and project administration.

Paper 3 [4] (Chapter 6) entitled "Analysis of Thin Carbon Reinforced Concrete Structures through Microtomography and Machine Learning" has been published in *Buildings* (Special Issue: *Research on the Performance of Non-metallic Reinforced Concrete*) by MDPI AG, Basel (CH).
Both, the author and Leonie Mester contributed equally to: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing – original draft, and visualization. All authors contributed to: writing – review & editing. Sven Klinkel and Hans-Gerd Maas are responsible for: resources, supervision, funding acquisition, and project administration.
As this article was written in a collaborative manner with Leonie Mester, the sections to which the author contributed are listed:
Abstract: 50 % (Section 6.0), Introduction: 50 % (Section 6.1), Specimens: 100 % (Section 6.2.1), Microtomography: 100 % (Section 6.2.2), AI-Based Segmentation: 100 % (Section 6.2.3), Roving Extraction: 100 % (Section 6.2.4), Concrete Cover c_0: 25 % (Section 6.15, volume binarization), Microtomography: 100 % (Section 6.3.1), Deep Learning: 100 % (Section 6.3.2), Roving Extraction: 100 % (Section 6.3.3), Conclusions: 50 % (Section 6.4).

[1] This CRediT statement differs from the one in the paper. This is the corrected version. Changes: Resources was moved from the author to Anette Eltner and Hans-Gerd Maas; project administration was moved from the author to Hans-Gerd Maas.

Acknowledgements

I would like to thank the many people whose support made this work possible.

Special thanks go to Prof. Dr. habil. Hans-Gerd Maas for his strong support throughout my time at the Institute of Photogrammetry and Remote Sensing. His guidance was crucial in making many important decisions.

I would also like to thank Junior Professor Dr. Anette Eltner, who was always willing to help me, especially when I started as a research assistant. Our discussions helped me to develop many new ideas that influenced my work.

I also want to thank Laura Camila Duran Vergara, M.Sc., for the wonderful time as an office colleague and friend. She made long days go by much faster.

I further want to thank Dr. Frank Liebold, without whom I might not have been able to follow this path. His role in helping me join the Institute of Photogrammetry and Remote Sensing where I could pursue this thesis was indispensable. He has been a valuable colleague at the institute and in the CRC/TRR280, helping me with various challenges.

I also appreciate my former team at descript (descript Solutions GmbH). Working with them significantly improved my programming skills, which were essential for my dissertation. Without the time spent with them, my thesis would not have been possible.

I am grateful to the Center for Information Services and High Performance Computing [Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)] at TUD Dresden University of Technology for providing its facilities for high throughput calculations.

The research work presented has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB/TRR 280; Project-ID: 417002380.

Abstract

Computed Tomography (CT) imaging provides invaluable insight into internal structures of objects and organisms, which is critical for applications ranging from materials science to medical diagnostics. In CT data, an object is represented by a 3D reconstruction that is generated by combining multiple 2D X-ray images taken from various angles around the object. Each voxel, a volumetric pixel, within the reconstructed volume represents a small cubic element, allowing for detailed spatial representation. To extract meaningful information from CT imaging data and facilitate analysis and interpretation, accurate segmentation of internal structures is essential. However, this can be challenging due to various artifacts introduced by the physics of a CT scan and the properties of the object being imaged.

This dissertation directly addresses this challenge by using deep learning techniques. Specifically, Convolutional Neural Networks (CNNs) are used for segmentation. However, they face the problem of limited training data. Data scarcity is addressed by data augmentation through the unsupervised generation of synthetic training data and the use of 2D and 3D data augmentation methods. A combination of these augmentation strategies allows for streamlining segmentation in voxel data and effectively addresses data scarcity. Essentially, the work aims to simplify training of CNNs, using minimal or no labeled data. To enhance accessibility to the results of this thesis, two user-friendly software solutions, unpAIred and AiSeg, have been developed. These platforms enable the generation of training data, data augmentation, as well as training, analysis, and application of CNNs.

This cumulative work first examines simpler but efficient conventional data augmentation methods, such as radiometric and geometric image manipulations, which are already widely used in literature. However, these methods are usually randomly applied and do not follow a specific order. The primary focus of the first paper is to investigate this approach and to develop both online and offline data augmentation pipelines that allow for systematic sequencing of these operations. Offline augmentation involves augmenting training data stored on a drive, while online augmentation is performed dynamically at runtime, just before images are fed to the CNN. It is successfully shown that random data augmentation methods are inferior to the new pipelines.

A careful comparison of 3D CNNs is then performed to identify optimal models for specific segmentation tasks, such as carbon and pore segmentation in CT scans of Carbon Reinforced Concrete (CRC). Through an evaluation of eight 3D CNN models on six datasets, tailored recommendations are provided for selecting the most effective model based on dataset characteristics. The analysis highlights the consistent performance of the 3D U-Net, one of the CNNs, and its residual variant, which excel at roving (a bundle of carbon fibers) and pore segmentation tasks.

Based on the augmentation pipelines and the results of the 3D CNN comparison, the pipelines are extended to 3D, specifically targeting the segmentation of carbon in CT scans of CRC. A comparative analysis of different 3D augmentation strategies, including both offline and online augmentation variants, provides insight into their effectiveness. While offline augmentation results in fewer artifacts, it can only segment rovings already present in the training data, while online augmentation is essential for effectively segmenting different types of rovings contained in CT scans. However, constraints such as limited diversity of the dataset and overly aggressive augmentation that resulted in segmentation artifacts require further investigation to address data scarcity.

Recognizing the need for a larger and more diverse dataset, this thesis extends the results of the three former papers by introducing a deep learning-based augmentation using a Generative Adversarial Network (GAN), called Contrastive Unpaired Translation (CUT), for synthetic training data generation. By combining the GAN with augmentation pipelines, semi-supervised and unsupervised end-to-end training methods are introduced and the successful generation of training data for 2D pore segmentation is demonstrated. However, challenges remain in achieving a stable 3D CUT implementation, which warrants further research and development efforts.

In summary, the results of this dissertation address the challenges of accurate CT data segmentation in materials science through deep learning techniques and novel 2D and 3D online and offline augmentation pipelines. By evaluating different 3D CNN models, tailored recommendations for specific segmentation tasks are provided. Furthermore, the exploration of deep learning-based augmentation using CUT shows promising results in the generating synthetic training data.

Future work will include the development of a stable implementation of a 3D CUT version, the exploration of new model architectures, and the development of sub-voxel accurate segmentation techniques. These have the potential for significant advances in segmentation in tomography data.

Zusammenfassung

Computertomographie (CT) bietet wertvolle Einblicke in die inneren Strukturen von Objekten und Organismen, was für Anwendungen von der Materialwissenschaft bis zur medizinischen Diagnostik von entscheidender Bedeutung ist. In CT-Daten ist ein Objekt durch eine 3D-Rekonstruktion dargestellt, die durch die Kombination mehrerer 2D-Röntgenbilder aus verschiedenen Winkeln um das Objekt herum erstellt wird. Jedes Voxel, ein Volumen Pixel, innerhalb des rekonstruierten Volumens stellt ein kleines kubisches Element dar und ermöglicht eine detaillierte räumliche Darstellung. Um aussagekräftige Informationen aus CT-Bilddaten zu extrahieren und eine Analyse und Interpretation zu ermöglichen, ist eine genaue Segmentierung der inneren Strukturen unerlässlich. Dies kann jedoch aufgrund verschiedener Artefakte, die durch die Physik eines CT-Scans und Eigenschaften des abgebildeten Objekts verursacht werden, eine Herausforderung darstellen.

Diese Dissertation befasst sich direkt mit dieser Herausforderung, indem sie Techniken des Deep Learnings einsetzt. Konkret werden für die Segmentierung Convolutional Neural Networks (CNNs) verwendet, welche jedoch mit dem Problem begrenzter Trainingsdaten konfrontiert sind. Der Datenknappheit wird dabei durch Datenerweiterung begegnet, indem unbeaufsichtigt synthetische Trainingsdaten erzeugt und 2D- und 3D-Augmentierungssmethoden eingesetzt werden. Eine Kombination dieser Vervielfältigungsstrategien erlaubt eine Vereinfachung der Segmentierung in Voxeldaten und behebt effektiv die Datenknappheit. Im Wesentlichen zielt diese Arbeit darauf ab, das Training von CNNs zu vereinfachen, wobei wenige oder gar keine gelabelten Daten benötigt werden. Um die Ergebnisse dieser Arbeit Forschenden zugänglicher zu machen, wurden zwei benutzerfreundliche Softwarelösungen, unpAIred und AiSeg, entwickelt. Diese ermöglichen die Generierung von Trainingsdaten, die Augmentierung sowie das Training, die Analyse und die Anwendung von CNNs.

In dieser kumulativen Arbeit werden zunächst einfachere, aber effiziente konventionelle Methoden zur Datenvervielfältigung untersucht, wie z. B. radiometrische und geometrische Bildmanipulationen, die bereits häufig in der Literatur verwendet werden. Diese Methoden werden jedoch in der Regel zufällig nacheinander angewandt und folgen keiner bestimmten Reihenfolge. Der Schwerpunkt des ersten Forschungsartikels liegt darin, diesen Ansatz zu untersuchen und sowohl Online- als auch Offline-Datenerweiterungspipelines zu entwickeln, die eine systematische Sequenzierung dieser Operationen ermöglichen. Bei der Offline Variante werden die auf der Festplatte gespeicherten Trainingsdaten vervielfältigt, während die Online-Erweiterung dynamisch zur Laufzeit erfolgt, kurz bevor die Bilder dem CNN gezeigt werden. Es wird erfolgreich gezeigt, dass eine zufällige Verkettung von geometrischen und radiometrischen Methoden den neuen Pipelines unterlegen ist.

Anschließend wird ein Vergleich von 3D-CNNs durchgeführt, um die optimalen Modelle für Segmentierungsaufgaben zu identifizieren, wie z.B. die Segmentierung von Carbonbewehrung und Luftporen in CT-Scans von carbonverstärktem Beton (CRC). Durch die Bewertung von acht 3D-CNN-Modellen auf sechs Datensätzen werden Empfehlungen für die Auswahl des genauesten Modells auf der Grundlage der Datensatzeigenschaften gegeben. Die Analyse unterstreicht die konstante Überlegenheit des 3D U-Nets, eines der CNNs, und seiner Residualversion bei Segmentierung von Rovings (Carbonfaserbündel) und Poren.

Aufbauend auf den 2D Augmentierungspipelines und den Ergebnissen des 3D-CNN-Vergleichs werden die Pipelines auf die dritte Dimension erweitert, um insbesondere die Segmentierung der Carbonbewehrung in CT-Scans von CRC zu ermöglichen. Eine vergleichende Analyse verschiedener 3D Augmentierungsstrategien, die sowohl Offline- als auch Online-Erweiterungsvarianten umfassen, gibt Aufschluss über deren Effektivität. Die Offline-Augmentierung führt zwar zu weniger Artefakten, kann aber nur Rovings segmentieren, die bereits in den Trainingsdaten vorhanden sind. Die Online-Augmentierung erweist sich hingegen als unerlässlich für die effektive Segmentierung von Carbon-Roving-Typen, die nicht im Datensatz enthalten sind. Einschränkungen wie die geringe Vielfalt des Datensatzes und eine zu aggressive Online-Datenerweiterung, die zu Segmentierungsartefakten führt, erfordern jedoch weitere

Methoden, um die Datenknappheit zu beheben.

In Anbetracht der Notwendigkeit eines größeren und vielfältigeren Datensatzes erweitert diese Arbeit die Ergebnisse der drei Forschungsartikel durch die Einführung einer auf Deep Learning basierenden Augmentierung, die ein Generative Adversarial Network (GAN), genannt Contrastive Unpaired Translation (CUT), zur Erzeugung synthetischer Trainingsdaten verwendet. Durch die Kombination des GANs mit den Augmentierungspipelines wird eine halbüberwachte Ende-zu-Ende-Trainingsmethode vorgestellt und die erfolgreiche Erzeugung von Trainingsdaten für die 2D-Porensegmentierung demonstriert. Es bestehen jedoch noch Herausforderungen bei der Implementierung einer stabilen 3D-CUT-Version, was weitere Forschungs- und Entwicklungsanstrengungen erfordert.

Zusammenfassend adressieren die Ergebnisse dieser Dissertation Herausforderungen der CT-Datensegmentierung in der Materialwissenschaft, die durch Deep-Learning-Techniken und neuartige 2D- und 3D-Online- und Offline-Augmentierungspipelines gelöst werden. Durch die Evaluierung verschiedener 3D-CNN-Modelle werden maßgeschneiderte Empfehlungen für spezifische Segmentierungsaufgaben gegeben. Darüber hinaus zeigen Untersuchungen zur Deep Learning basierten Augmentierung mit CUT vielversprechende Ergebnisse bei der Generierung synthetischer Trainingsdaten.

Zukünftige Arbeiten umfassen die Entwicklung einer stabilen Implementierung einer 3D-CUT-Version, die Erforschung neuer Modellarchitekturen und die Entwicklung von subvoxelgenauen Segmentierungstechniken. Diese haben das Potenzial für bedeutende Fortschritte bei der Segmentierung in Tomographiedaten.

Contents

Chapter 1

Introduction

Computed Tomography (CT) imaging has become an indispensable tool across various fields, offering insights into internal structures of objects and organisms. However, accurate segmentation in tomography data remains a significant challenge, critical for applications ranging from materials science to medical diagnostics and beyond. The ability to delineate and classify voxels within CT scans is fundamental to extracting meaningful information and enabling downstream analysis (Figure 1.1).

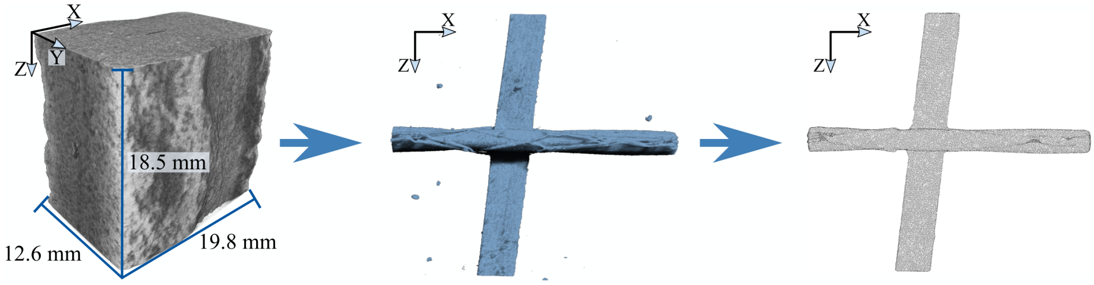


Figure 1.1: A carbon grid within a reconstruction of a carbon reinforced concrete was segmented and its surface represented as point cloud for further analysis.

This thesis delves into the landscape of tomography data segmentation through the lens of deep learning techniques. The research aims to reduce the scarcity of training data for neural networks by developing unsupervised data generation and offline and online data augmentation in 2D and 3D. Ultimately, this enables end-to-end training of Convolutional Neural Networks (CNNs) for segmentation in a semi-supervised and, under certain circumstances, even unsupervised manner.

The whole process from training to inference of a CNN (Figure 1.2) is simplified through the development of two user-friendly software solutions (unpAIred and AiSeg), making segmentation techniques more accessible and practical for researchers.

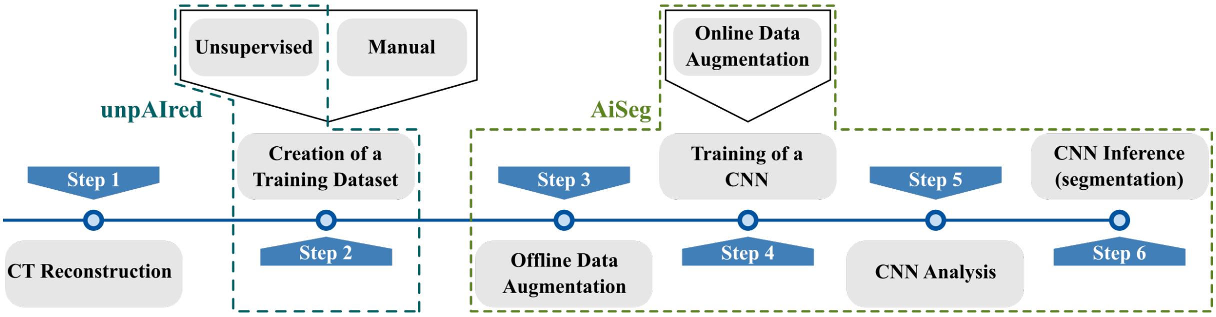


Figure 1.2: Process chain from CT reconstruction to inference of a CNN in the context of the developed software (unpAIred and AiSeg).

1.1 Thesis Structure

This work is a cumulative dissertation. It is based on a series of 3 scientific articles and an extension to the field of semi-supervised and unsupervised learning. Two articles have been published in peer-reviewed journals and one as a peer-reviewed conference publication. The author of this dissertation is the first author in each publication.

The structure of the work is described below, providing a deeper insight into the underlying relationships between the individual studies.

The next section of this chapter gives an overview of the integration of the work into the scientific context, whereby the necessity of deep learning strategies is discussed.

Following that, the two essential software packages developed in this thesis, AiSeg and unpAIred, are introduced in Chapter 2. They are a key factor in all publications, streamlining the training and analysis of deep neural networks for segmentation tasks. AiSeg addresses 2D and 3D CNN training and analysis (Figure 1.1 Step 3-6), while unpAIred addresses data scarcity by allowing unpaired image-to-image translation (Figure 1.1 Step 2).

Subsequently, Chapter 3 provides an introductory examination of the theoretical foundations from CT imaging to reconstruction, highlighting the key factors that influence image quality in CT reconstruction that motivate the need for deep learning.

Then, the first paper, "River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs", is presented in Chapter 4. It compares CNNs on the task of 2D river water segmentation and evaluates the performance of a new data augmentation pipeline for semantic segmentation with respect to an offline and online augmentation approach. An intensive test of the influence of individual methods is carried out in order to find optimal configurations for the pipelines.

The research then delves into 3D segmentation by comparing eight 3D CNN architectures to identify the most suitable with respect to certain dataset characteristics. The study, "A Comparative Study of Deep Architectures for Voxel Segmentation in Volume Images", emphasizes the importance of selecting a CNN based on the dataset attributes for optimal segmentation results and is presented in Chapter 5.

In Chapter 6, the research addresses a real-world problem: the segmentation of carbon grids from Carbon Reinforced Concrete (CRC). The third paper, "Analysis of Thin Carbon Reinforced Concrete Structures through Microtomography and Machine Learning", addresses challenges such as the disparity between training data and real data targeting CT scans of CRC with carbon grids, as well as the scarcity of training data. By extending the 2D data augmentation pipelines to 3D and leveraging insights from paper 1, along with selecting an appropriate 3D CNN model based on the findings of paper 2, the study demonstrates 3D semantic segmentation under realistic conditions.

Nevertheless, the preliminary results reveal weaknesses that cannot be remedied by basic data augmentation, such as radiometric and geometric manipulations, alone. To tackle data scarcity and potentially improve its variance, Chapter 7 delves into unsupervised and semi-supervised learning for image segmentation. In order to bridge the domain gap between synthetic and real training datasets for semantic segmentation, a Generative Adversarial Network (GAN) called Contrastive Unpaired Translation (CUT) [5] is adapted. The goal is to use CUT as a synthetic data generator to improve the generalization capabilities of segmentation models. Within that chapter, semi-supervised and unsupervised end-to-end trainings of 2D CNNs are performed using exclusively synthetically generated data by CUT and incorporating the results of Chapter 4 (paper 1). It demonstrates the feasibility and potential of a combination of supervised and unsupervised augmentation strategies for semantic segmentation.

Finally, the thesis concludes with a summary of the developments and an outlook on further research opportunities, especially regarding the extension of CUT to the third dimension.

The next section places the work in a scientific context. Machine learning methods and traditional approaches to image segmentation are presented, with a focus on their application to carbon and air segmentation. Research in 3D segmentation using neural networks is then presented. Finally, the importance of data augmentation is highlighted and compared to recent developments in the field.

1.2 Scientific Context

The dissertation was written in the scope of the Collaborative Research Center/Transregio 280, which focuses on material-minimized construction using CRC. With this material, only a very thin layer of concrete is required to cover the carbon reinforcement [6], which is why the quality control with regard to the position within a component is of particular interest. The porosity is also of great interest to further improve CRC and strengthen the construction component with respect to compression [7, 8]. Higher porosity tends to facilitate crack propagation, especially around pores. In compression scenarios, the primary goal is to minimize porosity and thus increase the strength of the concrete.

To achieve this goal, different rovings (bundles of carbon fibers) need to be segmented in different types of mortar and concrete, and the position of the rovings in the objects must be quantified by surface extraction. Furthermore, the porosity of some concrete specimens is to be investigated by pore segmentation. Due to the large number of components to be analyzed (currently more than 50 CT scans of CRC), a high degree of automation is required.

1.2.1 Developments in the Segmentation in Tomography Data

Segmentation in computed tomography is widely adopted [9–15] and algorithms have been categorized for example by Plaksyvyi et al. [16] into the following categories: threshold-based segmentation, region-based segmentation, edge-based segmentation, clustering-based segmentation, graph-based segmentation, active contour-based segmentation, and machine learning-based segmentation.

Generally, machine learning-based approaches outperform the other categories in terms of accuracy in 2D [16–19] and 3D [15, 20]. However, neural networks heavily rely on the existence of training data. Therefore, in domains where training datasets are rare, traditional, non-machine learning approaches can be a valuable alternative [21] and thus the choice of the algorithm depends on the use case and required accuracy.

In the domain of computed tomography, a CT reconstruction consist of grayscale images stacked on top of each other, where the intensity of a voxel, a volumetric pixel, relates mainly to the density of the scanned material at that position. However, a CT reconstruction contains a lot of image artifacts, which depends on various factors that will be explained in Chapter 3. Major artifacts are blurred edges and strong noise. Another, even more significant factor is that elements of the same density can have very different intensity values based on the location within an object and the reconstruction settings. Therefore, the applicability of the segmentation algorithms depends on scan parameters, the object to be scanned and the constituent that has to be segmented.

Segmentation of Carbon in Concrete

In order to carry out the segmentation of carbon, it is important to know which scan parameters are used and how precise a segmentation in CT data has to be. As this depends on the objective and the voxel size, a short example follows.

The company solidian GmbH produces thin textile reinforced concrete. They specify a minimum concrete coverage of 5 mm and a positional accuracy of ±0.2 mm [22] (in praxis, the concrete coverage is typically thicker (1.0 cm and more) [23, 24]). A micro-Computed Tomography (µ-CT) device achieves voxel sizes ranging from 1 to several micrometers. When scanning with a voxel size of 10 µm, this means that the positional accuracy is ±20 voxel. Consequently, the surface of a carbon segmentation can vary by a maximum of 20 voxels and still be within the positional accuracy requirements. With reconstruction dimensions of up to 3000 x 3000 x 3000 voxels for a CT reconstruction, a missegmentation with such a high difference is unrealistic assuming that a segmentation is checked for plausibility (see Figure 1.3b, 20 voxels are highlighted). Therefore, if the contrast is high enough and the carbon can be distinguished from the background (Figure 1.3a), the accuracy requirements allow for the use of traditional approaches that may not be as accurate, but require only a fraction of the time compared to the set up of a deep learning-based segmentation. However, standard algorithms require a high degree of manual adjustments and monitoring, which in turn slows down their application on a large scale.

In a study by Liebold et al. [25], where the author of this study co-authored, a large-scale in-situ CT experiment was conducted with a voxel size of 125 µm (Figure 1.3a). Due to the high energy used, a slice of the CT scan is rather homogeneous, i.e. the foreground (carbon) can be distinguished from the background (concrete) (Figure 1.3a). Thus, with a certain amount of effort, the carbon was accurately segmented using a non-local means filter, adaptive thresholding and a connected component analysis to estimate the concrete cover. In contrast, Chapter 6 shows that this approach is not sufficient for a µ-CT scan of CRC, where the reconstruction contains many components with similar gray values as carbon (Figure 1.3b). Once trained, a machine learning model's robustness makes it a faster and reusable method for segmenting carbon rovings.

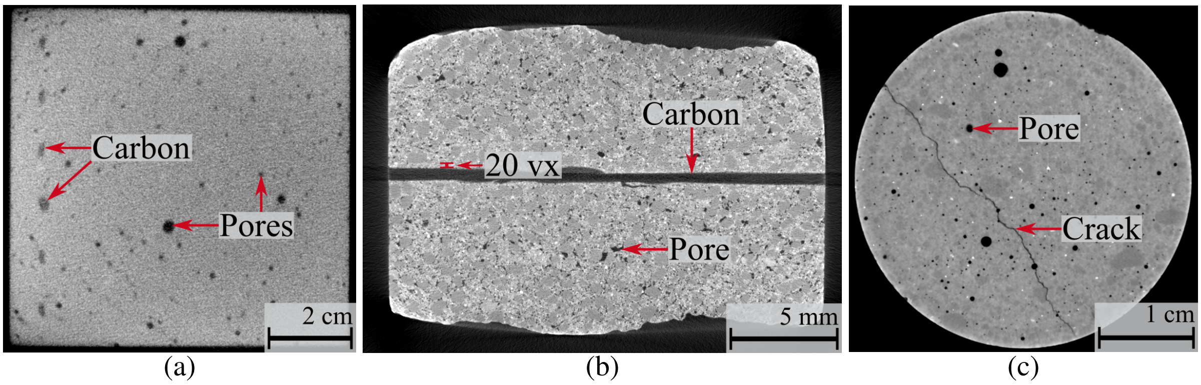


Figure 1.3: Examples of CT scans. **(a)**: CRC scanned with high energy (∼8 MeV) and a voxel size of 125 µm. Due to the high energy, only elements with larger density differences are visible. Concrete constituents cannot be distinguished. Small pores, due to the strong noise, appear almost similar to concrete. **(b)**: CRC scanned with low energy (∼100 keV) and a voxel size of 9.5 µm. Due to the smaller voxel size and lower energy, elements with similar density are distinguishable. Air and carbon are nearly indistinguishable due to smaller density differences and scan settings. **(c)**: Concrete scanned with low energy (∼100 keV) and a voxel size of 23.8 µm. Due to the reconstruction settings, air is clearly distinguishable from concrete.

Segmentation of Pores and Cracks

Since air is less dense than concrete, simple thresholding methods may be sufficient to segment air from concrete as shown in [7, 8, 26] (Figure 1.3c). However, due to the complex nature of CT scans, many studies report that thresholding approaches are often insufficient to segment pores or cracks (air) [14, 25, 27–29] (compare Figure 1.3a and b).

Barisin et al. [30] performed crack segmentation in CT reconstructions of concrete. They applied an edge-based segmentation algorithm called Hessian-based Percolatio (HP) [31] and a CNN called 3D U-Net [32] to CT scans to segment cracks. Both methods were successfully applied, while the HP algorithm required manual parameter adjustments and the 3D U-Net did not, highlighting the robustness of 3D CNNs for semantic segmentation. Notably, the 3D U-Net was trained only on semi-synthetic volumes [14], which underlines that training data does not always have to be generated manually.

1.2.2 3D Semantic Segmentation using Machine Learning

When 3D CT reconstructions are to be segmented using neural networks, the domain plays an important role in the choice of the CNN. In the context of 3D semantic segmentation with neural networks, there is a lack of studies that directly compare multiple 3D CNNs using the same training strategies. It is common that when evaluating a new 3D architecture, researchers compare their results with reported results of others, but do not use the same training strategies as in [33–36], or only compare with a few, but on the same dataset [37–39]. A primary reason is that 3D CNNs are computationally expensive and therefore performing a comparison is not practical [40].

Singh et al. [40] survey different 3D CNNs for semantic segmentation, among other tasks. They provide architectural information about different 3D CNNs and their results on different datasets, such as the Brain Tumor Segmentation Challenge (BraTS) [41], but do not elaborate on the training procedure or the hardware used. They also note that researchers often use interpolation methods to reduce the spatial dimension in order to fit full volumes into the memory of a Graphics Processing Unit (GPU), at the cost of significant information loss. The authors also conclude that GANs can be used for image synthesis, and explicitly cite the CycleGAN [42] as a reasonable approach for augmenting datasets with synthetic data.

He et al. [43] also provide an in-depth overview of 3D semantic segmentation for voxel classification. They report results on two different datasets, but also do not elaborate on the training strategies or the hardware used, which is essential since more Video Random Access Memory (VRAM) allows to use larger input dimensions without resizing and thus to increase the effective receptive field (i.e. the input dimensions) of a CNN, which directly increases its performance [40].

Chapter 5 aims to fill this gap by comparing eight different 3D CNNs for voxel classification (DenseVoxNet [33], HighResNet [34], Med3D [35], Residual 3D U-Net [36], 3D SkipDenseSeg [37], 3D U-Net [32], V-Net [38], LV-Net [39]). All CNNs are trained using the same datasets and training strategies, allowing the performance of these architectures to be evaluated. Within this study, 3 new datasets have been published (Fiber Segmentation Dataset [44], Carbon Rovings Segmentation Dataset [45], Concrete Pores Segmentation Dataset [46]), which are the first datasets focusing on the 3D segmentation of pores, fibers and carbon in CT scans of concrete.

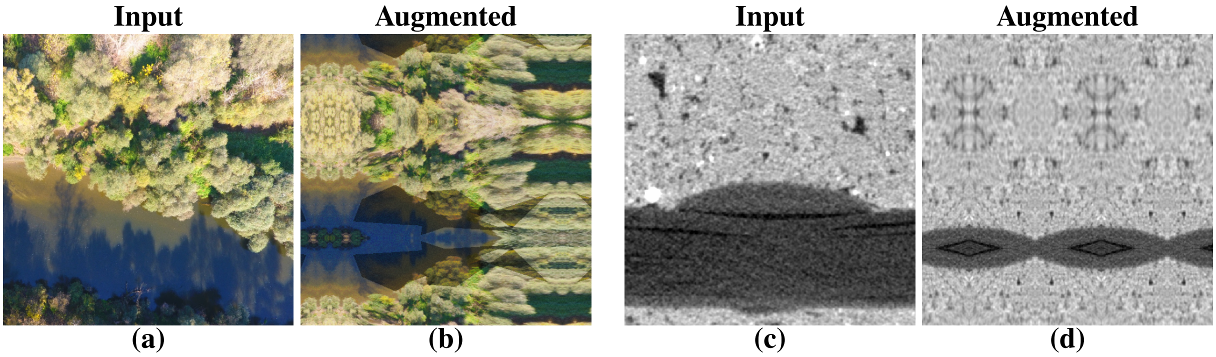
While CNNs have dominated 3D segmentation tasks, recent advances in transformers, particularly Vision Transformers (ViTs), have challenged their supremacy. Several studies have shown that ViTs [47, 48], often combined with CNNs [49, 50], provide superior accuracy with optimized resource usage [51]. ViTs excel at handling noisy or augmented images due to their self-attention mechanism, providing holistic information processing. However, CNNs outperform ViTs in scenarios with limited training data, showing better generalization capabilities and accuracy [52]. Since very few training datasets are available in this study, ViTs are not part of this research. Furthermore, the study *ConvNets Match Vision Transformers at Scale* by Smith et al. [53] (Google DeepMind), although not in the domain of image segmentation but classification, challenges the presumed superiority of ViTs over CNNs on large datasets. Normalizer-Free Network (NFNet) models [54], pre-trained on a large dataset of nearly 3 billion images, show comparable performance to ViTs when fine-tuned on ImageNet (90.4 % top-1 accuracy). They suggest that ConvNets can perform comparably to ViTs on this task if evaluated fairly.

1.2.3 Data Augmentation

In the training of neural networks, data augmentation plays a crucial role in enhancing the model's performance and generalization capabilities. By diversifying the training dataset through techniques such as rotation, flipping, scaling, cropping, and adding noise, neural networks become more robust to variations in input data. Additionally, data augmentation effectively expands the size of the training dataset without the need for collecting additional labeled data. This regularization technique helps prevent overfitting, a phenomenon where the model learns to memorize specific features of the training data excessively, leading to poor performance on unseen data. By encouraging the model to learn more invariant and generalizable features, data augmentation mitigates overfitting and improves the model's ability to generalize to new data. Moreover, data augmentation can mitigate biases present in the original dataset, address class imbalances, and improve the model's adaptability to new environments or datasets [55–58]. Shorten et al. [59] classify data augmentation into basic image processing and deep learning approaches.

Basic Data Augmentation

Conventional methods remain predominant in the domain of semantic segmentation due to their simplicity, efficiency, and established effectiveness [58–60]. By leveraging basic image manipulations such as rotations, translations, and others, researchers can effectively augment existing datasets without the need for complex computational frameworks or extensive computational resources. One of the main benefits is that these methods ensure shape consistency between the augmented images and their target masks (ground truth). However, an arbitrary concatenation of these methods can lead to overly augmented images that do not contain meaningful information (Figure 1.4) and is therefore counterproductive. Therefore, in Chapter 4, augmentation pipelines are developed that overcome these problems. These pipelines allow to chain several conventional methods while keeping the image content meaningful.



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

Figure 1.4: Examples of overly augmented images. Too many augmentations were concatenated so that the outputs do no more contain meaningful information. **(a)** and **(b)**: Input and augmented image of a drone image. **(c)** and **(d)**: Input and augmented image of a µ-CT slice of a carbon roving in concrete.

Deep Learning-based Data Augmentation

Unlike traditional approaches, deep learning techniques learn representations directly from the data, enabling tailored transformations that optimize performance. In a study by Qin et al. [61], a hybrid conventional and deep learning method has been presented. They used Deep Reinforcement Learning (DRL) to identify the optimal conventional augmentation method for improving segmentation results. DRL is a method in which an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. In this context, the reward is an improved DICE score, which measures the equality between the ground truth and the segmentation, based on the choice of augmentation (the environment). However, training two networks simultaneously is computationally intensive and time consuming, especially for 3D segmentation tasks.

Another promising approach is unpaired image-to-image translation. In this augmentation method, GANs are utilized to translate images from one domain to another without paired examples, potentially eliminating the need to manually generate training datasets. The term paired refers to training data consisting of input images and their corresponding pixel-wise annotations. GANs, such as CycleGAN [42], CUT [5], or MUNIT [62] can generate diverse synthetic images, thereby expanding existing or generating entirely synthetic training dataset and thus can improve model generalization for semantic segmentation tasks. However, these methods often encounter the domain gap problem, where the generated images do not fully capture the diversity of the target domain, rendering the generated datasets inadequate. In addition, there may be challenges in maintaining the correct shape when translating between domains, since changing the shape in segmentation tasks would result in non-correlated input-target pairs.

Despite these challenges, recent studies have shown promising results. For instance, Imbusch et al. [63] successfully used CUT to generate images for semantic segmentation by adapting synthetic images of rendered 3D scenes to the real domain, effectively minimizing the domain gap. Similarly, another study by Myers et al. [64] used a modified CycleGAN to create a realistic annotated synthetic dataset for wheat head segmentation, achieving high DICE scores on both internal and external datasets. These findings suggest that such approaches can be extended beyond specific applications to other domains such as computed tomography. Therefore, Chapter 7 of this thesis demonstrates the application of CUT to generate synthetic training data for segmentation in CT scans.

Chapter 2

Developed Software Solutions: AiSeg and unpAIred

In the field of deep learning, the development of reproducible, user-friendly tools that democratize access to complex neural network architectures and analysis techniques is paramount to empowering researchers and practitioners of varying expertise, thereby fostering innovation and collaboration, and lowering barriers to entry. Unfortunately, research software is often not written according to software development standards or is not published at all. [65, 66]

This chapter introduces the two main software packages that were developed and open-sourced during this thesis, namely AiSeg and unpAIred. These software tools are designed to streamline the training and analysis of deep neural networks in a flexible and versatile manner by following best practices. Both packages were programmed using Python, JavaScript, Hypertext Markup Language (HTML), and Cascading Style Sheets (CSS), and feature a web-based Graphical User Interface (GUI) hosted on a local server, eliminating the need for command line interfaces or custom code modifications. The easy to use-nature is complemented by code being best-practice conform and tutorials that are provided on the project pages. In essence, the projects accelerate reproducible scientific work through their GUI, the database connection and the logging function.

AiSeg (https://gitlab.com/fra-wa/aiseg) is a comprehensive solution for training, analyzing, and evaluating 2D and 3D Convolutional Neural Networks (CNNs) for image and volume segmentation tasks. Its interface allows non-programmers to easily navigate through the various functionalities, such as model analysis, comparison or logged information, without having to delve into complicated code bases.

Complementing AiSeg, unpAIred (https://gitlab.com/fra-wa/unpaired) addresses the challenge of non-existent or scarce training data by unpaired image-to-image translation to reduce the labor-intensive effort of creating training datasets. The software allows to convert synthetic training data from a source domain into realistic looking data of a target domain, thereby reducing the domain gap between synthetic and real datasets for pixel and voxel classification.

In order to simplify the maintenance and interaction with both software solutions, it was ensured that the projects follow the same design patterns in terms of architecture, frontend (user interface and user experience, see Figure 2.1) and backend (server-side components).

This chapter first describes the underlying software architecture, followed by a demonstration of the straightforward installation process. Thereafter, the two software solutions are introduced. At the end of this chapter, the current limitations of both software are presented.

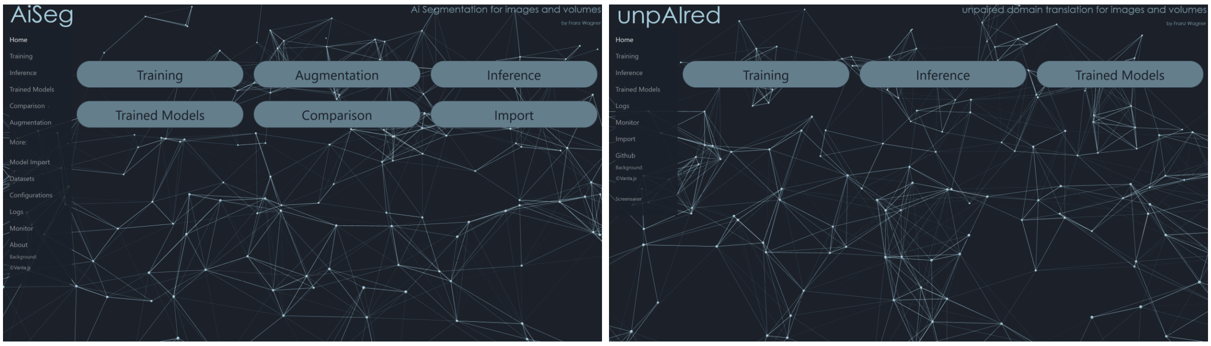


Figure 2.1: Main menus of AiSeg (left) and unpAIred (right).

2.1 Software Design

Both AiSeg and unpAIred are developed using Django, a Python web framework known for its rapid development and clean design principles [67]. In addition, both rely heavily on PyTorch [68] and NumPy [69] among many other packages that can be found on the project packages. The software architecture, shown in Figure 2.2, describes the interaction between users and the software:

A *User* can navigate through the web pages, served by the backend, and can interact with the frontend (1). When a *User* submits data or clicks on a link, an Hypertext Transfer Protocol (HTTP) request is sent to the Django server (2), where the request is handled by a *View*. Each *View* represents a page of the website and contains some logic that handles the request. In general, the request is either a link, that redirects to another *View* or it contains some data that is submitted via a *Form*. A *Form* ensures that the submitted data meets criteria such as required fields or correct formatting (3). If any validation errors occur, feedback is sent back to the frontend, where the *User* can correct the input. Forms can also store validated data in *Models* (4), which define the structure and behavior of the *Database*. *Models* essentially perform Create, Read, Update, Delete (CRUD) operations (5). If the input into the *Database* is valid, the *View* receives a "valid" flag from the *Form* and typically starts a background *Task* (6), such as data augmentation or neural network training. These *Tasks* interact with *Models* to store relevant information (7), such as training metrics. Furthermore, a *Model* called *LogFile* is instantiated for each task, which receives the logged output of its process (7) and stores it in the *Database* and as a text file, allowing multiple processes to run in parallel. For convenience, the contents of a *LogFile* instance can be monitored live via a *Monitoring View*. Here the *User* can follow the output of each subprocess live and has the option to kill it via an input *Form*.

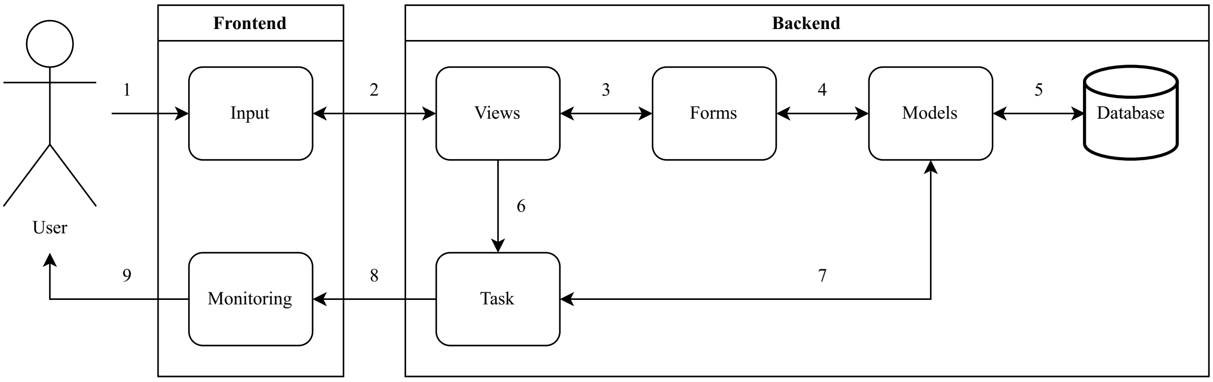


Figure 2.2: Software design of AiSeg and unpAIred: A user submits data (1) via a form to the backend (2). The data is processed at the related view (3), validated at a form, then stored in the database through a model class (4, 5). After validation, a task is started in the background (6) that interacts with database models (7). The logged output is streamlined to a monitor view (8) for progress monitoring (9).

2.2 Installation

Transitioning from discovering an open-source research project to initiating the first training phase can be a time-consuming endeavor. This is largely due to the absence of comprehensive documentation, potential bugs, or unforeseen code behaviors that hinder researchers from replicating specific elements of a research paper. Such hurdles not only slow down progress but can also obstruct the wider community from effectively utilizing the software.

AiSeg and unpAIred aim to avoid these pitfalls as they offer a seamless installation process, requiring only a double-click to install and another double-click to execute (on Windows; Linux necessitates two terminal calls). The only prerequisites are Python 3.9 and an Nvidia Graphics Processing Unit (GPU). From downloading the project to starting a first training, the entire process takes only 7 minutes[1], assuming the presence of a training dataset.

2.3 AiSeg

In recent years, there has been an explosion of open-source projects focusing on image or volume segmentation using neural networks[2]. To the best of the authors knowledge, only the Chinese framework called PaddleSeg [70] offers support for more 2D neural networks (>40) for image segmentation than AiSeg (35) (https://gitlab.com/fra-wa/aiseg). However, AiSeg stands out as the sole repository featuring a user-friendly interface while also supporting both 2D (35) and 3D CNNs (9), and includes an advanced augmentation algorithm for offline and online data augmentation. The augmentation algorithm efficiently increases variance through a fast chaining of augmentation operations, which will be introduced in Chapter 4. Since the results of online augmentation cannot be monitored by an operator as in offline augmentation, AiSeg allows previewing possible generations with images from the dataset to validate the selected parameters.

Furthermore, AiSeg adheres to various best practices outlined in [66], [71], and the "Python Enhancement Proposal (PEP) 8 - Style Guide for Python Code" [72], which become critical as research code scales, as seen with AiSeg and unpAIred (together exceeding 53 000 lines of code). In summary, key considerations include:

- Adherence to PEP 8 style guide in order to write clear, consistent, readable, and well-documented code that improves comprehension, maintenance, and collaboration among developers.
- Utilizing an Integrated Development Environment (IDE) like PyCharm [73] to streamline coding processes, boost productivity, and simplify debugging.
- Robust unit testing to ensure that the code, i.e. functions and their interactions, behave as intended [74].
- Performance optimization, including:
  - Utilizing NumPy arrays [69] for improved efficiency in image processing.
  - Leveraging GPU-accelerated computing with libraries like PyTorch [68].
  - Employing multicore processing for enhanced performance [75].
- Following a modular design that facilitates seamless integration of new augmentation functions, neural networks, or database models into the software without disrupting the existing codebase.

[1] 100 000 MBit connection

[2] Examples: PaddleSeg, Semantic Segmentation on PyTorch, TorchSeg, Semantic Segmentation in PyTorch, Segmentation Models, Pytorch Medical Segmentation, Segmentation models 3D Zoo, Pytorch 3D Medical Image Semantic Segmentation, pytorch-3dunet, and many more

2.4 unpAIred

In contrast to image segmentation, to the best of the authors knowledge, there are currently no repositories dedicated to domain-to-domain translation that encompass multiple models. Despite the lack of maintenance, these projects tend to provide better code quality due to the complexity of their architectures, which requires accurate programming.[3] The aim of unpAIred (https://gitlab.com/fra-wa/unpaired) is to unify suitable unpaired image-to-image translation models into a single software platform and expand their capabilities into 3D. Ultimately, it functions as a synthetic data generation tool designed to complement AiSeg. Currently, unpAIred features the Contrastive Unpaired Translation (CUT) model introduced by Park et al. [5]. Notably, unpAIred stands out in the image translation domain for its inclusion of a graphical user interface, an aspect often lacking in similar software.

Unlike CNNs, Generative Adversarial Networks (GANs) involve a more intricate training process. They often require a customized training approach that involves adopting specific architectures and implementing tailored training methodologies. While unpAIred also adheres to the modular principles, integrating additional domain translation GANs into the software therefore proves to be more challenging than adding a new neural network to AiSeg. However, expanding the software with other models is possible without disrupting the existing codebase.

2.5 Limitations

AiSeg requires the training datasets to follow a specific folder structure. Although this requirement is well documented on the project page, it can be perceived as a limitation, as users must conform to this structure and may need to preprocess their training data accordingly. However, this structure is intended so that new users are forced to prepare their data in a standardized way, which allows AiSeg to extract some metadata from the datasets.

Furthermore, AiSeg does currently not incorporate vision transformers. As of 2021, transformers have gained prominence in image segmentation tasks. However, during AiSeg's early stages (2020-2021), vision transformers were not readily available and were not widely adopted for image segmentation. The introduction of the first transformer-based segmentation architecture (SETR) in 2021 marked a significant development in this area [76, 77]. Nevertheless, owing to AiSeg's modular design, it can be extended to incorporate transformers, thus mitigating this limitation.

Currently, unpAIred exclusively accommodates a single domain-to-domain translation model (CUT). While this may evolve over time, it limits the intended comparable nature in comparison to AiSeg.

Although unpAIred contains large and tested parts of AiSeg's code, such as the data augmentation, many tests are still missing at the current early stage. However, this is part of future work to guarantee a (nearly) bug-free application.

unpAIred does not yet have a working version of a 3D CUT for volume-to-volume translation. Furthermore, such a version requires a considerable amount of Video Random Access Memory (VRAM). Training a 3D CUT model on a 128^3 voxel volume requires approximately 48 GB of VRAM. This requirement mandates access to either a supercomputer or a powerful workstation. To put this in perspective, a 512^2 pixel image requires only about 11 GB of VRAM.

[3] Examples: CUT, DP-GAN, DiscoGAN, DualGAN, StarGAN , UNIT, MUNIT,

Chapter 3

Factors Affecting Image Quality in Computed Tomography

The quality of a Computed Tomography (CT) reconstruction is important for the choice of the segmentation technique and data preparation. Hence, it is essential to understand the artifacts that can occur during CT scanning and its subsequent reconstruction process, as they may require machine learning techniques depending on the object being scanned. The main factors contributing to quality degradation will be described in this section:

1. X-ray tube and focal spot
2. Beam hardening
3. Absorption, scattering and pairing
4. The X-ray detector
5. The calibration of a CT scanner
6. The reconstruction algorithm

Some factors contribute to the same type of artifacts. Some example images are therefore shown multiple times to emphasize this.

3.1 From CT Scan to Reconstruction

CT and micro-Computed Tomography (µ-CT) scanners are used to examine the internal characteristics of objects. Such a device, schematically depicted in Figure 3.1, consists of an X-ray source (1), a rotating sample platform (4), and a detector (6). During the scanning process, the sample (3) is rotated until it completes a full 360° rotation. At each rotation step, the object is exposed to polychromatic X-ray radiation (2) and the detector captures a projection image (5). The detector measures the remaining X-ray photons, which correspond to the attenuation of the X-rays due to the interaction with matter. Subsequently, the acquired X-ray images (7) are used to reconstruct a virtual 3D representation (8) of the object.

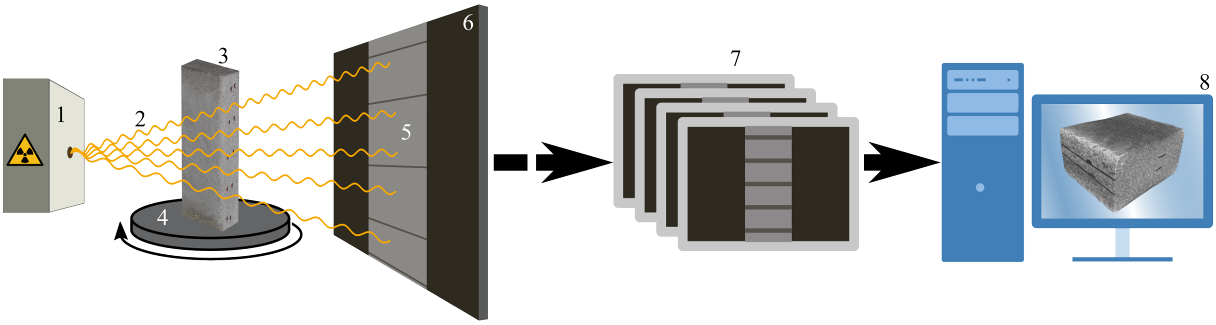


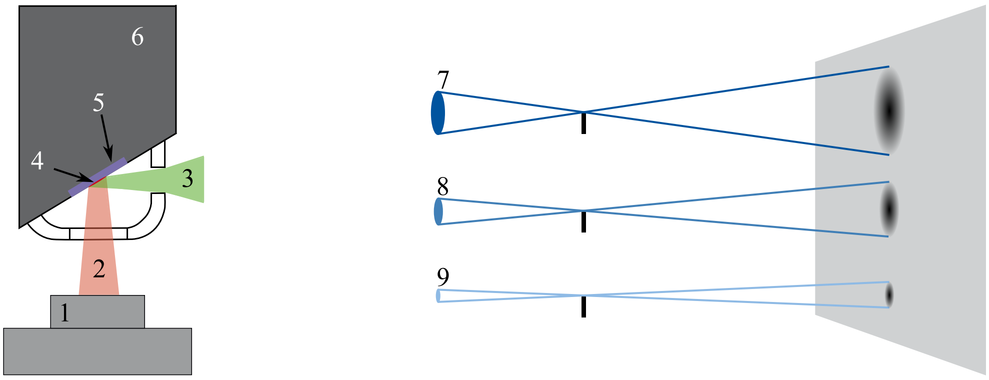
Figure 3.1: CT scan to reconstruction. 1: X-ray source, 2: polychromatic X-rays, 3: sample, 4: rotating sample plate, 5: projection on 6: detector, 7: stack of projection images, 8: PC with reconstruction. The left part of the image (1 to 6) is adapted with permission from [4]

3.2 X-ray Tube and Focal Spot

X-ray radiation is produced within the CT device's X-ray tube (Figure 3.2a), comprising a cathode (1) and an anode (6) within a vacuum chamber. The cathode, typically made of tungsten wire, emits electrons when a high voltage is applied, generating a focused electron beam (2). The anode, usually a tungsten target (5), interacts with this beam and produces X-ray photons (3). Variations in an X-ray tube output can result in fluctuations in the detected signal, contributing to image noise.

The focal spot in a CT system relates to the small area, where the focused electron beam hits the anode. The hit area determines the size and shape of the focal spot (4 in Figure 3.2).

The focal spot size in a CT system impacts spatial resolution and contrast. A smaller focal spot yields higher spatial resolution, enabling clearer visualization of fine details as depicted in Figure 3.2b. Conversely, a larger focal spot may lead to reduced spatial resolution and geometric blurring, resulting in less sharp edges. [78]



(a)

(b)

Figure 3.2: Scheme of a focal spot **(a)** and of its influence on the spatial resolution on the detector **(b)**. 1: cathode; 2: electron beam; 3: X-rays (cone beam); 4: focal spot (red); 5: tungsten target (purple); 6: anode (heat-conducting copper block with tungsten target); 7 to 9: scheme of a large to small focal spot.

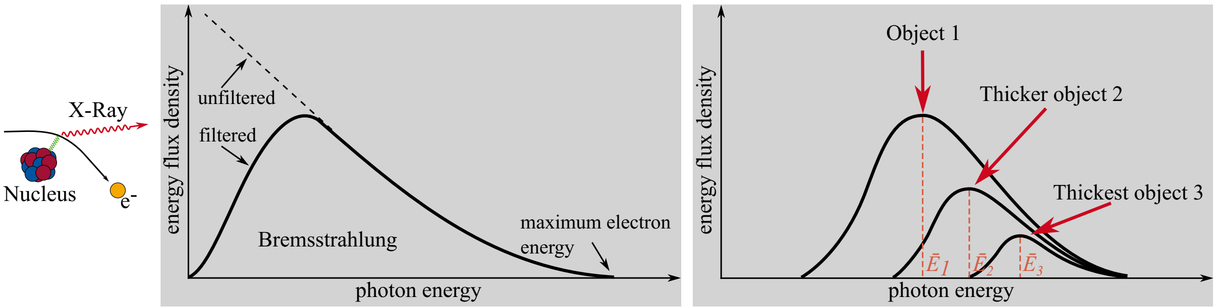
3.3 Beam Hardening

The X-ray photons produced at the anode are primarily generated by Bremsstrahlung [79, 80].

Bremsstrahlung occurs when high-speed electrons decelerate near the positively charged nucleus of the target atoms. Due to the deceleration, the electron loses energy and X-ray photons are emitted. The energy of the emitted photons depends on factors such as initial electron energy and distance from the nucleus (Figure 3.3a).

Bremsstrahlung is the major contributor to the continuous X-ray spectrum (Figure 3.3b). In a theoretical unfiltered spectrum, all photons within the X-ray tube would emerge from the system without being altered. In reality, however, electrons penetrate the target material to some depth because the emitted photons interact with many atoms and can't easily leave the material. This interaction filters out lower energy photons. [81]

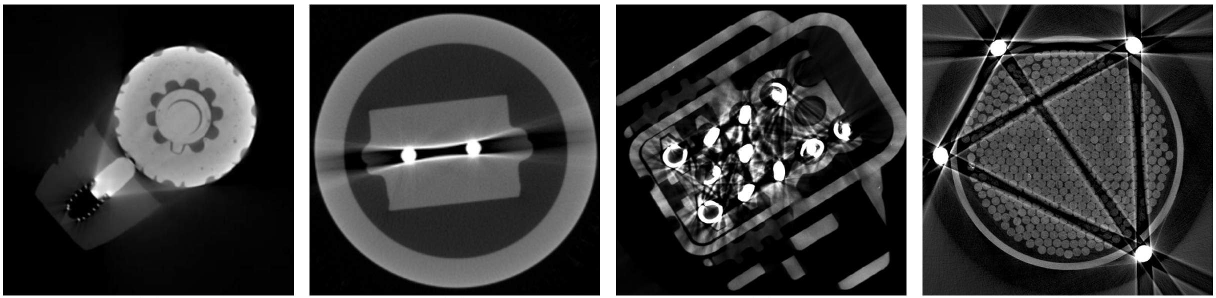
When the X-ray beam passes through an object, beam hardening occurs. As a result, photons of lower energy are attenuated, leading to an increase in the average energy of the X-ray beam, causing it to become "harder" (Figure 3.3c). The X-ray beam's spectrum shifts towards higher energies, which leads to artifacts, such as cupping, streaking, and inaccuracies in CT images, particularly in areas with higher density materials. [82]



(a) **(b)** **(c)**

Figure 3.3: **(a)** Bremsstrahlung: A rapid, high-energy electron is decelerated and deflected by the electromagnetic force (green), leading to the generation of X-rays (red). **(b)** X-ray spectrum: Schematic of an X-ray spectrum created by Bremsstrahlung and some atoms characteristic radiation. **(c)** Beam hardening: X-rays of lower energies are attenuated while traversing through an object.

In cupping, the central area of the reconstructed image appears darker compared to the surrounding periphery (Figure 3.4a). Streaking artifacts manifest as streak-like patterns within the image, resulting from irregularities in X-ray attenuation (Figure 3.4b-d).



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

Figure 3.4: Examples of artifacts introduced by beam hardening. **(a)**: Cupping. **(b)** to **(e)**: Examples of streaking effects. Images provided by Dr. Marian Willner (X-AID [83])

3.4 Absorption, Scattering and Pairing

X-rays interact with matter differently based on their energy levels. Key interactions include photoelectric absorption, rayleigh scattering, compton scattering, and pair production. Typical CT scans in this dissertation were performed at energies from 100 keV to 8 MeV.

Photoelectric absorption involves an X-ray photon transferring its energy to an inner-shell electron of an atom, causing ionization and absorption of the photon (Figure 3.5a). It is material dependent and more likely at lower X-ray energies and in materials with higher atomic numbers (e.g. water: up to about 100 keV). Excessive absorption in certain regions can cause artifacts in the reconstructed image, such as streaks or shadows.

Rayleigh scattering occurs when an X-ray interacts with an entire atom, briefly oscillating it before emitting a scattered photon with the same energy but different direction (Figure 3.5b). It occurs at lower X-ray energies (e.g. water: up to about 100 keV) and contributes minimally to attenuation in terms of noise and contrast reduction.

Compton scattering happens when an X-ray photon interacts with an outer-shell electron, transferring some energy to it, causing scattering at a different angle with reduced energy (Figure 3.5c). It is present in all CT scans in this thesis, as it occurs between low and very high X-ray energies (e.g. water: up to about 10 MeV), being less dependent on atomic number than photoelectric absorption. This effect causes streaks, shadows, reduces the contrast, and is one of the primary causes of the beam hardening phenomenon.

Pair production becomes significant at energies above 1 022 keV. Here, high-energy photons interact with a nucleus' electric field, transforming into an electron-positron pair (Figure 3.5d). This effect introduces noise, streaks and shadows [84, 85]

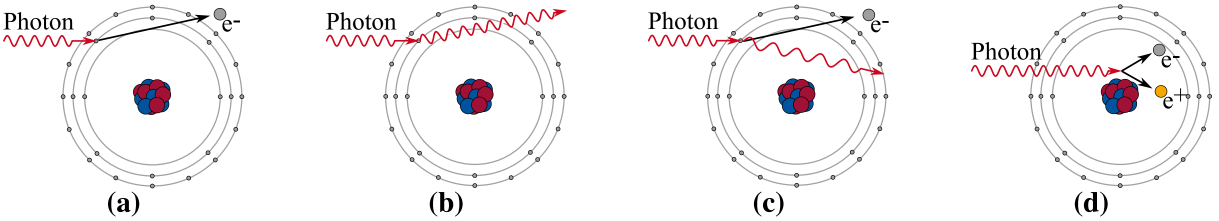


Figure 3.5: **(a)** Photoabsorption: Absorption of the incoming single X-ray photon and ejection of an electron. **(b)** Rayleigh scattering: The X-ray photon changes its direction. **(c)** Compton scattering: The incoming photon ejects an electron, releasing a new photon with less energy. **(d)** Pair production: The incoming photon is destroyed, releasing an electron and a positron.

Figure 3.6 shows the practical impact of the effects. All artifacts contribute to the occurrence of streaks and the increase in the noise level.

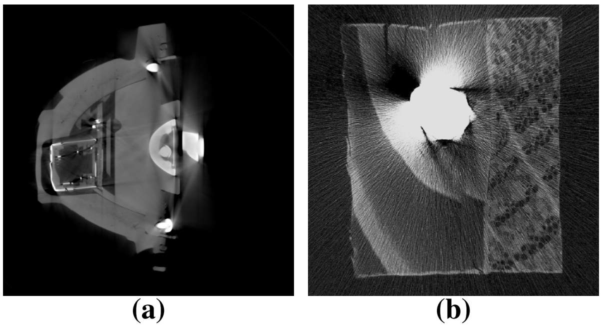


Figure 3.6: Examples of artifacts introduced by absorption, scattering, and pairing. **(a)**: Scattering produces streaks, similar to beam hardening but weaker. **(b)**: All artifacts can contribute to the noise level, which is likely at very low or very high energies due to photon starvation or pairing. Images provided by Dr. Marian Willner (X-AID [83])

3.5 X-ray Detector

In the scope of the Collaborative Research Center/Transregio 280, scintillation and semiconductor detectors are used [86, 87]. Semiconductor detectors can cause spectral distortions leading to artifacts like streaks, rings or noise (Figure 3.7) due to inconsistent attenuation measurements [88]. In contrast, scintillation detectors may introduce noise, blurring, and image distortions due to properties of scintillation crystals and light collection mechanisms, as well as variations in light output and energy resolution, impacting spatial resolution and image quality [89].

As with a conventional camera, a sensor used in computed tomography does not have a perfectly flat plane. This effect is a rarely considered error source, but it also contributes to the introduction of ring, and streak artifacts. [90]

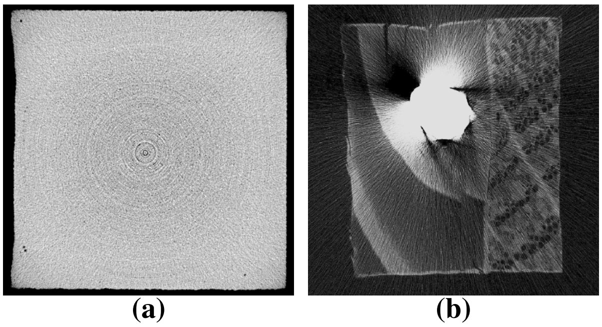


Figure 3.7: Examples of artifacts introduced by the detector. (**a**): Erroneous pixels can cause ring artifacts. (**b**): Insufficient detector signal due to excessive attenuation. Images provided by Dr. Marian Willner (X-AID [83])

3.6 Geometric Calibration

Geometrical calibration in CT is crucial for ensuring accurate reconstruction of images by precisely aligning the geometry of the X-ray source, object, and detector. An incorrect calibration can lead to various artifacts in the reconstruction, including geometric distortions and blurring (Figure 3.8). Accurate geometrical calibration indirectly contributes to minimizing artifacts such as streaking and cupping by ensuring proper alignment of the imaging components. [91, 92]

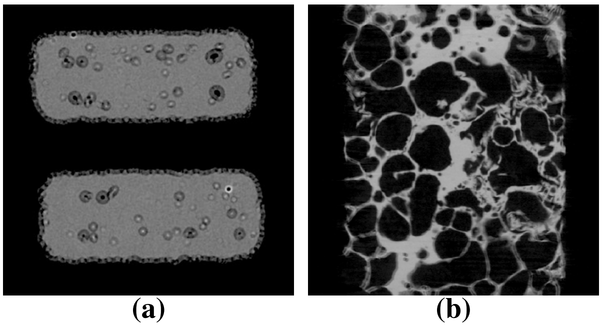


Figure 3.8: Examples of artifacts caused by poor calibration or axis misalignment. (**a**): Axis of rotation is offset. (**b**): Rotation axis is tilted. Images provided by Dr. Marian Willner (X-AID [83])

3.7 Reconstruction Algorithm

The reconstruction software X-AID from MITOS GmbH [83] is used in this study, which supports the filtered backpropagation algorithm of Feldkamp-David-Kress (FDK) under license from the Institute of Photogrammetry and Remote Sensing (TU Dresden).

The FDK algorithm is a commonly used method for reconstructing 3D images from CT data. One of its main advantages is its ability to preserve spatial resolution well and thus capturing fine details in the reconstructed images. Additionally, FDK is computationally efficient, making it suitable for rapid reconstruction tasks. [93]

However, FDK may exhibit also weaknesses that affect image quality. It can produce artifacts, such as ring artifacts (Figure 3.9a) and streaks (Figure 3.9b). Furthermore, similar to other filtered back projection algorithms, FDK is sensitive to noise, which can degrade image quality, especially in low-dose imaging scenarios (Figure 3.9c). [94, 95]

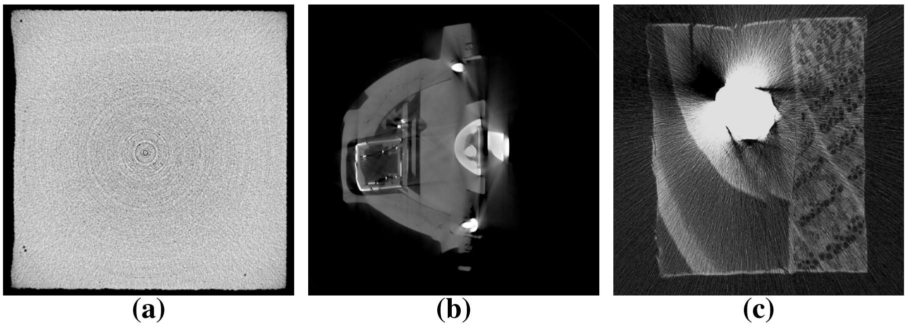


Figure 3.9: Examples of artifacts introduced by the reconstruction algorithm. **(a)**: Ring artifacts. **(b)**: Streaking artifacts. **(c)**: Noise amplification. Images provided by Dr. Marian Willner (X-AID [83])

3.8 Artifact corrections

To correct the artifacts, a reconstruction software usually offers different correction algorithms. X-AID features a correction for each of the factors mentioned in this chapter. These are shown in Figure 3.10. Although the reconstruction software can mitigate some artifacts, it cannot completely eliminate these imperfections (Figure 3.10a – f, i).

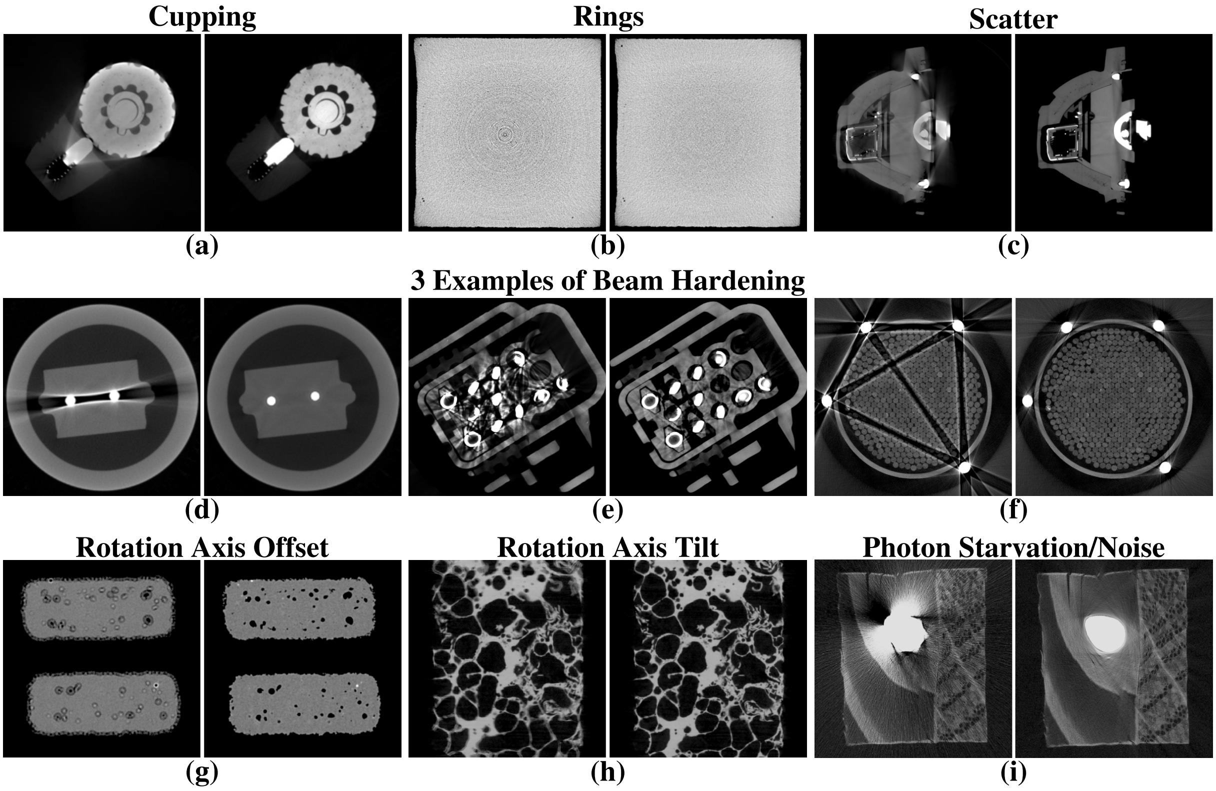


Figure 3.10: Examples of artifacts and their correction with X-AID. Some artifacts cannot be fully corrected, such as cupping **(a)**, rings **(b)**, scattering **(c)**, streaks **(d)** to **(f)**, and photon starvation/noise **(i)**. The success of the correction for cupping, streaks and noise depends strongly on the object to be scanned (example: **(d)** and **(e)**). Images provided by Dr. Marian Willner (X-AID [83])

Chapter 4

On the Development of Augmentation Pipelines for Image Segmentation

Chapter 4 was published in the "International Journal of Applied Earth Observation and Geoinformation", Volume 119, 2023 (ISSN 1569-8432) as: **River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs** [2]

Authors: Franz Wagner[1], Anette Eltner[2], Hans-Gerd Maas[1]

[1] Chair of Photogrammetry, TU Dresden, Dresden, Germany
[2] Junior Professorship in Geosensor Systems, TU Dresden, Dresden, Germany

Publication history: Submitted on February 3rd, 2023; revised on April 5th, 2023; accepted on April 11th, 2023; published on May 3rd, 2023.

Full reference: Wagner, F.; Eltner, A.; Maas, H.-G. (2023). River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs in: *International Journal of Applied Earth Observation and Geoinformation* 119, 103305, DOI: 10.1016/j.jag.2023.103305

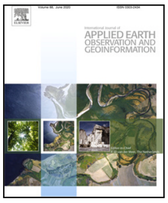
Contribution:
Following the "CRediT" guidelines by Brand et al. [1] (https://doi.org/10.1087/20150211), the author is responsible for: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing – original draft, and visualization. All authors contributed to: writing – review & editing. Anette Eltner and Hans-Gerd Maas are responsible for: resources, supervision, and funding acquisition. Hans-Gerd Maas is responsible for: project administration.

International Journal of Applied Earth Observation and Geoinformation 119 (2023) 103305

Contents lists available at ScienceDirect

International Journal of Applied Earth Observation and Geoinformation

journal homepage: www.elsevier.com/locate/jag



River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs

Franz Wagner[a,*], Anette Eltner[b], Hans-Gerd Maas[a]

[a] *Chair of Photogrammetry, TU Dresden, Dresden, Germany*
[b] *Junior Professorship in Geosensor Systems, TU Dresden, Dresden, Germany*

ARTICLE INFO

Keywords:
Deep learning
Segmentation
Water segmentation
Augmentation
CNN comparison

ABSTRACT

To obtain reliable water segmentations from image data for real-time monitoring of river water levels, a comparison of 32 convolutional neural networks was performed. They were trained on a new river water segmentation dataset consisting of 1128 images. To prevent overfitting, two methods using offline and online augmentation were developed to improve the variance. It was found that offline augmentation is superior on fewer data, while online augmentation is advantageous for a larger dataset (such as Cityscapes).

The network comparison showed that U-Net performs best on the water segmentation dataset when using an ResNeXt 50 encoding network pre-trained on ImageNet. It achieves an intersection over union (*IoU*) of 0.91 without augmentation, 0.98 with offline augmentation and 0.93 with the online augmentation method. The authors have applied the algorithms for online and offline augmentation to Cityscapes to verify the applicability of the strategies to other datasets. The mean *IoU* is 0.86 without augmentation, 0.86 with offline augmentation and 0.87 with online augmentation. Only online augmentation could prevent overfitting on Cityscapes.

1. Introduction

In recent years, flooding disasters have become more frequent, causing loss of life and billions of dollars in damage. In order to monitor, predict and eventually respond to such events, real-time water levels are a crucial observation. Typically, water level measurement in rivers is conducted by conventional gauges, such as floating and pressure gauges. However, conventional gauges can be costly and are thus mainly concentrated on major rivers, while smaller rivers, which can be prone to flash flood events, often have a rather sparse gauging infrastructure. An option to support conventional gauges may be created on the basis of a network of low-cost cameras, combined with software for precise and reliable water level detection such as in Eltner et al. (2021), where a convolutional neural network (CNN) is used to segment the water body of a river. Their idea is to project the boundaries of the segmentation into a 3D Structure from Motion model to obtain the metric water level, as done in Elias et al. (2019) and Elias and Maas (2022). By using a Raspberry Pi camera, they have proven that such a system can accurately measure the water level. However, as this approach targets a single scene, the trained CNN cannot be used in other locations as it was only trained with images of this river. Blanch et al. (2022b) attempt to extend this idea using a well-generalized CNN to segment the water in different areas.

This paper focuses on developing such a generalized neural network using a limited dataset, which is a common problem in environmental sciences. In order to develop a suitable segmentation pipeline, **three main tasks** arise, which are addressed in this work.

The dataset: First, a new high quality dataset for the training of a CNN is created. The dataset called RIWA (River Water Segmentation Dataset; Blanch et al., 2022a; link to RIWA) represents a first version of pixel-wise binary river water segmentation with resolutions up to 1536 × 1536 pixels. It contains 789 training images, 228 validation images and 111 test images, which are a combination of manually labeled smartphone, UAV (unoccupied aerial vehicle) and DSLR images of rivers as well as suitable images of the Water Segmentation Dataset (Liang et al., 2020).

Overfitting: Overfitting occurs when a neural network learns to perfectly model the features of the training data, but not the features that occur in other samples such as the validation and test data. When this happens, it will not perform well on new, unseen images (Shorten and Khoshgoftaar, 2019). Since CNNs are heavily reliant on large datasets to prevent overfitting, two new augmentation algorithms for offline and online augmentation are developed to increase the size of the dataset. Applying augmentation techniques before the training procedure is called offline data augmentation. If the augmentation is applied during the training, it is called online augmentation.

[*] Corresponding author.
E-mail address: franz.wagner@tu-dresden.de (F. Wagner).

https://doi.org/10.1016/j.jag.2023.103305
Received 3 February 2023; Received in revised form 5 April 2023; Accepted 11 April 2023
Available online 3 May 2023

During this study, we conducted an extensive test to estimate optimal augmentation methods. We compared the results with intuitive parameters to find out whether an extensive parameter analysis is necessary or if carefully considered estimates would be sufficient.

The developed offline and online augmentation algorithms are compared to a training without augmentation. They are intensively tested on the RIWA dataset. To verify applicability of the developed methods to other datasets, we applied the algorithms to the well-known Cityscapes dataset (pixel-wise semantic urban scene understanding). The high quality Cityscapes version consists of 2975 training, 500 validation and 1525 test images and is divided into 40 different classes such as cars, persons, traffic lights and more (Cordts et al., 2016).

Which CNN to choose?: Third, the best performing CNN is determined, since there are many published architectures. Some publications claim that their developed architecture performs better at specific tasks than other state-of-the-art networks, such as the U-Net 3+ (Huang et al., 2020) against the DeepLab 3+ (Chen et al., 2018), which has yet to be confirmed on the domain of water segmentation. Therefore, a comparison of 32 different CNNs (91 when considering different feature extraction networks) is performed to ensure a segmentation of best quality. A list of all networks and their references can be found in Appendix A.

To conduct all tests, a new software (link to GitLab) is developed and made freely available. It is capable of interactively training 2D and 3D CNNs, augment data with the aforementioned augmentation methods, analyze single networks, compare multiple networks and apply trained CNNs to new data. To make a rough estimate of the training duration, we approximated the time spent as if we trained using only a single A100-SXM4 (40 GB) GPU, resulting in approximately 643 days of training. If we had used a single RTX 3090 (24 GB) GPU, the training duration would have increased to more than six years (2286 days).

In summary, this paper has three major contributions:

1. It provides a comparison of different neural networks to segment river water bodies
2. It provides a comparison of offline vs. online augmentation for two datasets
3. It presents an easy-to-use software for training, analyzing and using segmentation CNNs with different augmentation strategies

2. Methods

This section gives an overview of the data preparation, augmentation, the two training strategies developed besides the standard training (without augmentation), the networks and metrics used as well as the experimental design.

2.1. Data preparation

Images and masks contained in many datasets are non-squared and/or of arbitrary size. Therefore, a dataset is automatically preprocessed to generate squared input samples. If an image is smaller than the user-defined input size, it is enlarged under the following condition: The minimum dimension in height h or width w must be equal to the input size s ($min(w, h) = s$). If the width or height is larger than the input size, the image and its ground truth are divided into several sub patches, each having the expected squared input size. Example: if an image of size 1500 × 900 pixels is divided into patches of 350 × 350 pixels, this will result in 15 sub-images as shown in Fig. 1. The offset is evenly distributed among the overlapping areas.

The newly created images represent the training data passed to a CNN during training without augmentation. In offline augmentation, the original input images and masks are first augmented and then processed using the described procedure. In contrast, in online augmentation, the decomposed sub-images are manipulated. When such a sub-image is loaded into memory, it is augmented and then passed to the CNN, which is the main difference between the two approaches.

2.2. Augmentation

2.2.1. Strategies to prevent overfitting

Regarding image classification tasks, augmentation in deep learning is already thoroughly investigated (Santana et al., 2022; Shorten and Khoshgoftaar, 2019; Wambugu et al., 2021). Since this study focuses on precise pixel-wise segmentation, the more complex algorithms like meta-learning algorithms such as AutoAugment (Cubuk et al., 2019a) and its successors (RandAugment Cubuk et al., 2019b, KeepAugment Gong et al., 2020) or neural style transfer (Gatys et al., 2015) are not suitable as they cannot be easily adapted. The -Augment algorithms each use a neural network that learns advantageous augmentation strategies for classification or object detection tasks. KeepAugment, for example, learns which part of an image is most important and tries to keep that area in each augmented example. Neural style transfer uses a style painting, e.g., Starry Night by Van Gogh, and the original image and applies the style of the painting to the original image, which results most likely in small geometric distortions which is considered to be bad for precise segmentation.

Therefore, image-based radiometric and geometric augmentation is chosen in this study. Most deep learning libraries have implemented some basic image transformations, which are usually limited to combinations of flipping, rotating, scaling and cropping (Buslaev et al., 2020; Paszke et al., 2019) and are widely used (Alam et al., 2021; Xu et al., 2023). In the albumentations library (Buslaev et al., 2020), significantly more augmentation techniques have been added for 2D images. Nevertheless, there is no consensus on the best general augmentation strategy (Shorten and Khoshgoftaar, 2019) and the size of generated training dataset depends on the operators experience (Yang et al., 2022). In addition, the impact on different networks, the performance of an online approach and how many augmentations should be concatenated and applied on a single image remain unknown.

2.2.2. Domain specific augmentations

It is well known that the choice of appropriate augmentations depends on the domain. For example, in certain classification problems, rotating the number 6 by 180 degrees would result in a 9, but it is still labeled as a 6. Such domain-specific properties must be considered to determine optimal augmentation methods without extensive testing. With regards to the RIWA dataset, we assume that geometric augmentations are slightly more important than radiometric operations. Geometric augmentations offer the ability to create different acquisition angles by simply rotating or tilting the image. Different camera distortions can be calculated or image sizes can be varied via simple cropping or resizing operations. In addition, the shape of the rivers can be augmented by squeezing the images or using a combination of all methods. However, rotating a river image beyond 45 degrees or inverting it vertically is unlikely to be useful, since the cameras are set for water level measurement and the images are not rotated during post-processing. The radiometric space is also expected to vary widely owning to different seasons, weather (sun, clouds, fog, heavy rain, storms), different river beds, climate zones, environmental disasters, reflection, or man-made pollution (Fig. 2). For instance, the color of the river may change during floods as a result of sediment, turning it brown. However, the water can also be clear, so that the riverbed appears almost black if it is a deep river or the texture of the riverbed becomes well visible in shallow waters (Fig. 2). The river may also be rippled due to rapids or storms resulting in white foam. Other color changes can be caused by algae, which can turn the river green to blue–green (green algae) or red (Rhyncholacis clavigera).

Similar considerations should be made for the Cityscapes dataset. Here, the focus is on detecting cars, people, facades, or bicycles, which requires the network to emphasize shape over color, since, for example, purple or pink cars are rare but do exist. On the other hand, strong geometric manipulations of car shapes are likely to be counterproductive. Therefore, radiometric operations are expected to have a greater effect than geometric methods.

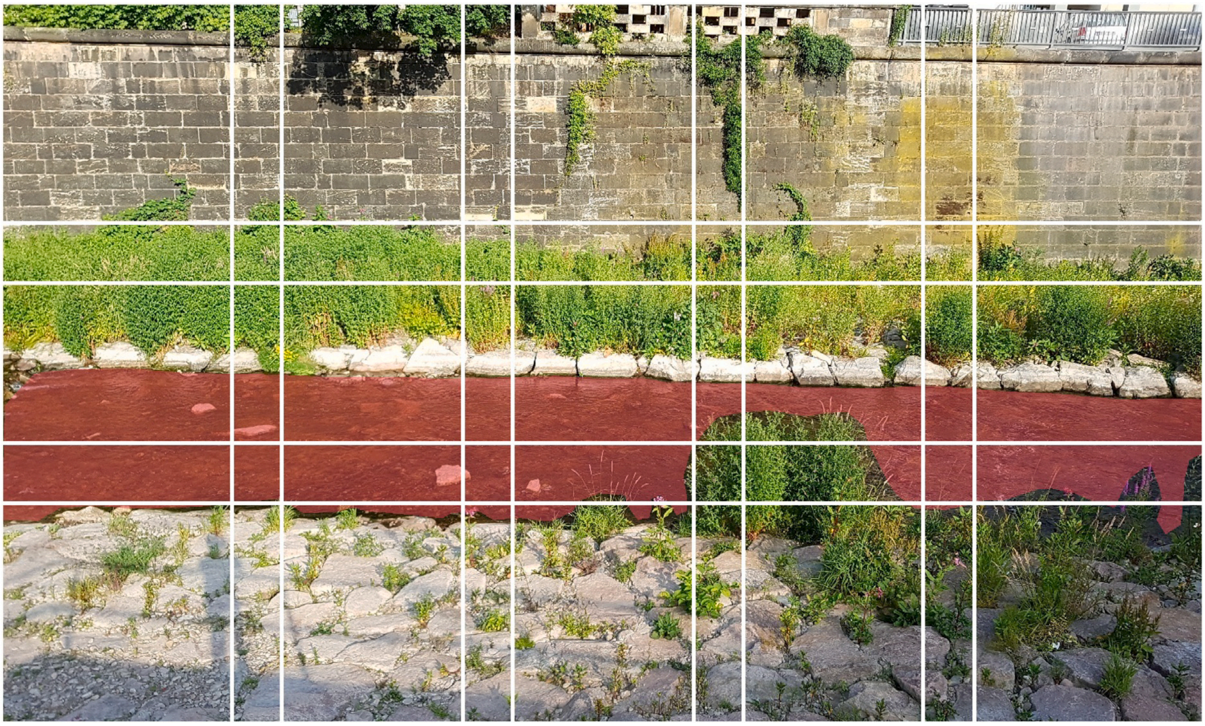


Fig. 1. Subdivision of an image (1500 × 900 pixel) into nine 350 × 350 pixel patches (white).

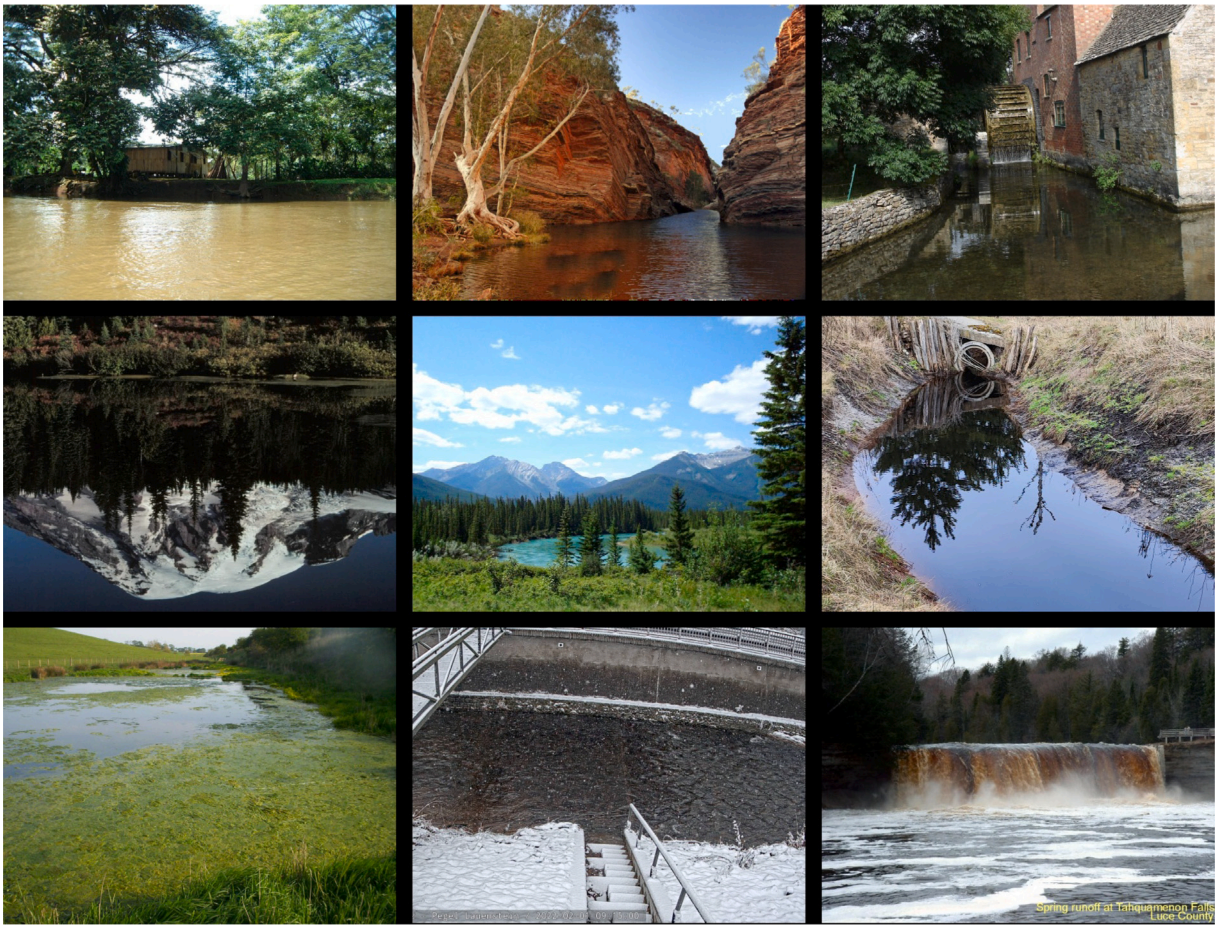


Fig. 2. Examples of rivers representing different water colors from images included in the RIWA dataset.

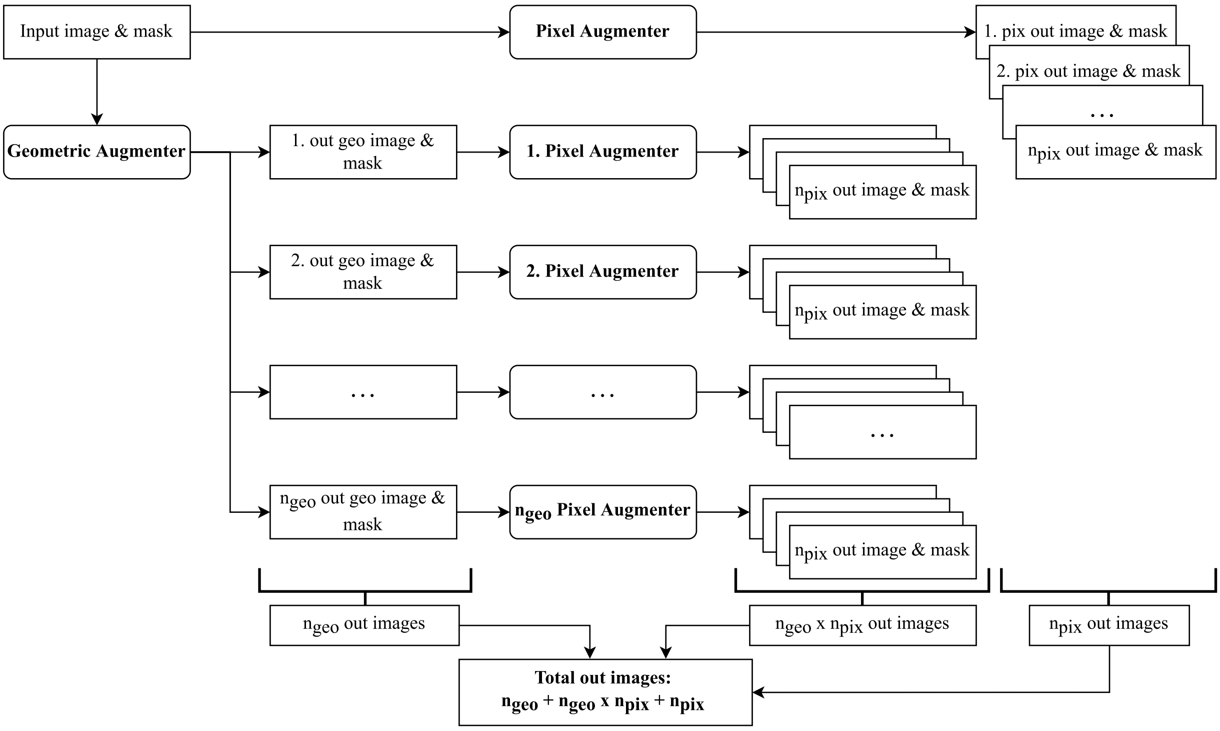


Fig. 3. Offline augmentation scheme of an input image and mask (top left). It is augmented by the Pixel Augmenter (top, middle) and the Geometric Augmenter. The output samples of the Geometric Augmenter are also processed by the Pixel Augmenter (center).

2.2.3. *Offline augmentation*

Acquiring annotated training data of high quality can be very labor intensive and time consuming. The term augmentation refers to methods, which manipulate existing annotated data in such a way that it provides new information to the neural network, for example flipping an image and its ground truth from left to right. Applying such manipulations before the training has started is called offline augmentation (Shorten and Khoshgoftaar, 2019). Hereby, the operator retains control over the dataset. The augmented images and masks can be inspected by the operator beforehand to check if the augmentations are reasonable.

One disadvantage of this approach is the much larger required storage space. There are many possible combinations of different augmentations for one image and it is impossible to store them all. Example: The software currently allows up to 36 different offline augmentations Appendix B. Let us assume that three augmentations are randomly selected and chained. This would lead to a total number of $36!/(36-3)! = 42840$ variations for one image. To reduce the amount of stored images while maintaining a large variety, the implemented offline augmentation uses a different, less random strategy. The pipeline as shown in Fig. 3 consists of two augmenters: a geometric augmenter containing 15 different operations (elastic distortion, flip (height, width), grid distortion, grid shuffle, optical distortion, random crop, resize, rotation, squeeze (height, width), tilt (backward, forward, left, right)) and a pixel augmenter, containing 21 radiometric operations (adaptive histogram, blur, brightness, channel shuffle, color to hsv, contrast manipulation, fog, histogram normalization, iso noise, noise, non-local means smoothing, rain, random erasing, rgb shift, shadow, sharpen, snow, solarize, sun flair, to gray, to sepia). Examples of all operations can be found in Appendix B.

The algorithm is developed as follows: After specifying a maximum number of augmentations per image (n_{total}), the geometric augmenter is instructed to create $n_{geo} = \lceil 1/2 * n_{total} \rceil$ geometrically transformed versions of the input image. Afterwards, the pixel augmenter creates $n_{pix} = n_{total} - n_{geo}$ versions of every created geometric image as well as n_{pix} versions of the input image (see Fig. 3).

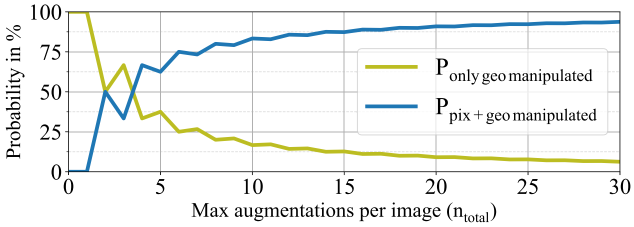


Fig. 4. Probability distribution of an image getting augmented only using a geometric method (green) and a combined approach (geometric and radiometric augmentation) (blue) with respect to the maximum number of augmentations per image.

As this results in many more images than the set n_{total}, only n_{total} randomly selected images are stored. The final distribution between only geometrically manipulated images and pixel transformed images for one input sample with respect to the set n_{total} is shown in Fig. 4. It is calculated by:

$$P_{geo} = \frac{n_{geo}}{n_{geo} * n_{pix} + n_{pix}} \tag{1}$$

$$P_{pix} = 1 - P_{geo} \tag{2}$$

The larger the number of augmentations, the more likely it is that each augmented image will contain a pixel manipulation. The idea is, that geometric augmentations are directly preserved in the output images of the pixel augmenter. However, since some pixel manipulations such as channel swapping or conversion from RGB to HSV (hue saturation value) color space are strong augmentations with respect to the image channels, the original color space should be preserved by keeping $P_{geo} * n_{total}$ geometric transformed images. It is up to the operator to set a domain dependent n_{total} based on considerations explained in Section 2.2.2. An example of such augmentations is shown in Fig. 5.

In conclusion, this approach allows for a wide variety of augmented images while keeping the dataset size at an acceptable level.

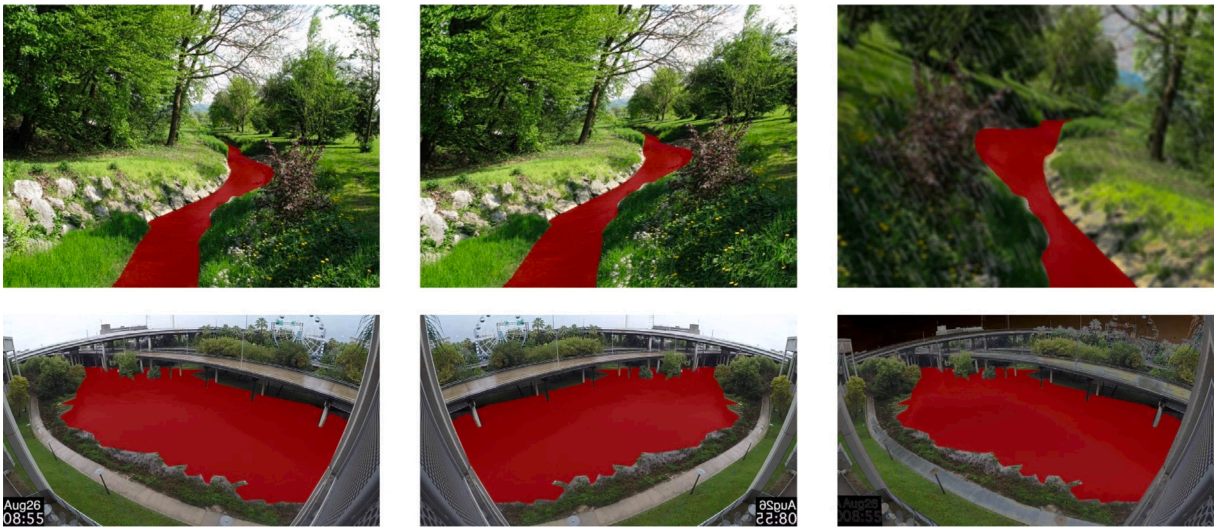


Fig. 5. Two offline augmentation examples: left: original image with ground truth as overlay; middle: output of a geometric transformation; right: output of a geometric and pixel augmentation.

2.2.4. Online augmentation

In contrast, online augmentation does not require more storage space as the operations are executed on the fly (Shorten and Khoshgoftaar, 2019). During the training phase, as explained in Section 2.1, sliced sub-images are randomly loaded into memory. Each image can now be augmented with a probability (P_{global}) defined by the operator.

If the augmentation is triggered for an image, the image itself and its ground truth, are processed by the developed pipeline. First, it randomly selects a user-specified number of augmentations (n_{max}) from a list of 27 supported operations. They are reduced from 36 to 27 since the tilt, squeeze and flip operations are combined to random versions of each. The grayscale operations, such as non local means filter and adaptive histogram, are not supported. The image and its ground truth are then augmented by applying every selected method, one after another.

To further extend the customization, every augmentation method itself has its own activation probability ($P_{operation}$, set by the user). Therefore, even if the pipeline is active, it is possible that no or up to n_{max} operations are concatenated and applied to a single training image and its associated ground truth.

Since some geometric operations require extrapolation of the image content in order to recreate the expected input size (example: elastic distortion, see Fig. 6, top right), the order of the operations is important as well. Therefore, the online augmentation allows the use of different policies named *random*, *geo first*, *pix first*, *geo only* and *pix only*. *Random* simply selects random operations. *Geo first* mimics the offline augmentation, while *pix first* behaves inversely. *Geo only* uses only random geometric operations, while *pix only* uses randomly selected pixel manipulating algorithms.

Applying this technique during the training in the main thread would slow down the process considerably. Therefore, multiple workers are loading the data in simultaneous threads to avoid blocking computation code. A batch is returned to the main thread on demand without losing time in the main process.

The online augmentation method allows for the creation of a theoretically much larger dataset. However, to achieve best possible results, a CNN using online augmentation has to be trained for more epochs than without augmentation. On the other hand, this approach iterates significantly faster than using offline augmentation as only the original training data is stored on the device.

2.3. Networks

The augmentation techniques are intensively tested on the RIWA dataset and tested afterwards on the Cityscapes dataset. To compare the results, 32 different architectures are implemented and trained with each technique. Some Networks are able to exchange their encoding part, also called backbone, in which case it is pre-trained on ImageNet (Deng et al., 2009). In the following, only the best performing backbones are used to visualize results (if multiple backbones are supported). A complete list with all encoders can be found on the project page.

All networks are trained using mostly the same hyperparameters: The input size is fixed to 512 × 512 pixels. No weighting regarding eventual class imbalances is applied. To evaluate the networks performance during training and validation, a loss value is used. It is calculated by the loss function. Regarding the RIWA dataset, the binary version of the Cross Entropy Loss is used:

$$L = -(y \log(p) + (1 - y) \log(1 - p)) \tag{3}$$

When facing a multi-class problem (Cityscapes), the default cross entropy loss is chosen:

$$L = -\sum_{c=1}^{C} y_{o,c} \log(p_{o,c}) \tag{4}$$

where: L = Loss value
p = Probability: observation o is of class c
y = ∈[0, 1]; indicator if classification c is correct
C = Number of classes

The optimizer, which uses the loss value to adjust the networks weights, is set to Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001. As one goal is to process the training data as fast as possible, the batch sizes differ. Due to the size of some networks, in some instances a batch size of less than 16 had to be set. In these cases Group Norm is used, which achieves similar and more stable performance compared to Batch Norm on small batch sizes (≤16) (Wu and He, 2018). No hyper-parameter tuning regarding the validation data is performed, allowing the use of the validation dataset as a second testing dataset. The random seeds, a value with which a random number generator is initialized, of all libraries regarding the CPU and GPU are fixed in this study. Therefore, with the exception of some non-deterministic CUDA implementations (Morin and Willetts, 2020), every training can be re-produced.

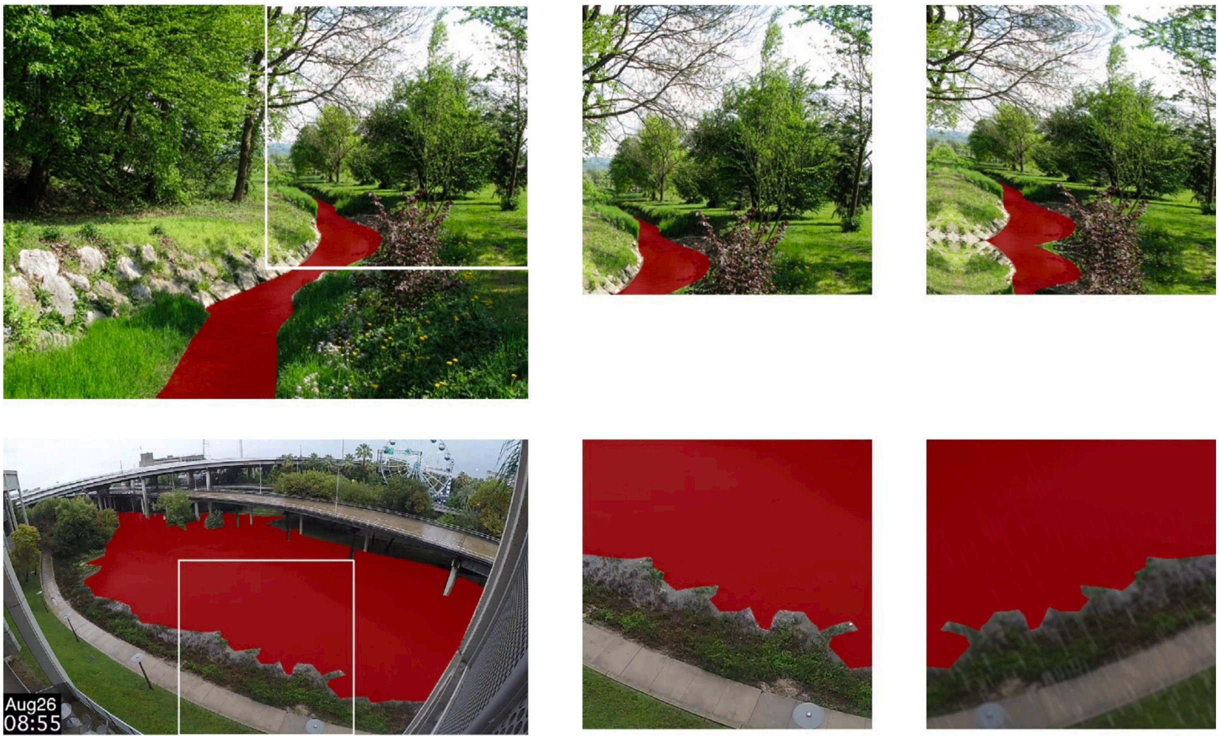


Fig. 6. Two online augmentation examples: left: original image with ground truth as overlay, white bounding box: 512 × 512 patch passed to the pipeline; middle: extracted 512 × 512 patch; right: output of the augmentation pipeline, serves as input to the network.

2.4. Training and metrics

Since the training of neural networks may not be familiar, a brief summary of the training procedure follows:

1. Iterate over the training dataset.
2. Calculate the training loss and accuracy.
3. Adjust the network based on the training loss.
4. Iterate over the validation dataset.
5. Calculate the **validation loss** and **validation accuracy**.
6. Repeat until the validation loss does not decrease anymore.

Where:

- The loss value is a measure of the distance between ground truth and prediction.
- The accuracy refers to the ratio of correctly classified pixels to all pixels in an image.
- The validation loss is not used to update the network.
- The validation loss and accuracy are estimates of how well the neural network performs on unseen data.
- The number of repetitions is called epochs.

Since the validation data is usually used to tune hyperparameters such as batch size, normalization technique, optimizer, learning rate, etc., another dataset, the test dataset, is required. The test dataset is processed after the training is completed using the best trained network in terms of the validation loss.

However, loss and accuracy are not good evaluation measures by themselves. Suppose a 100 × 100 pixel image contains a 10 × 10 pixel square of foreground. If the network predicts all pixels to be background, accuracy is still 99%, and without weighting, loss is low, falsely indicating good performance to the operator. Therefore, the **Intersection-over-Union** (***IoU***, also called Jaccard Index) (Taha and Hanbury, 2015) is used as the testing metric. It is defined as the area of the intersection divided by the area of union. Considering a binary segmentation problem (RIWA), the IoU is calculated by:

$$IoU = \frac{TP}{TP + FP + FN} \tag{5}$$

where: $TP =$ True Positives
$FP =$ False Positives
$FN =$ False Negatives

If a multiclass problem with n classes is investigated (i.e. Cityscapes), this metric refers to the weighted mean IoU ($mIoU$) which is calculated by summarizing the IoU per class (IoU_c), adjusted by a weight (w_c). The w_c is calculated by dividing the sum of all pixels in the dataset, representing a class c, by all pixels of the dataset:

$$w_c = \frac{\sum_i^t Pixels_{c_i}}{\sum_i^t Pixels_i} \tag{6}$$

where: i = image
t = total images

The weighting is introduced here to address class imbalances regarding the final score that are typical for segmentation tasks. Therefore the $mIoU$ represents a global rather than a class-biased performance. It is calculated as follows:

$$mIoU = \sum_{c=1}^{n} w_c * IoU_c \tag{7}$$

where: n = all classes

It is important to state, that every pixel of both datasets relates to exactly one ground truth class. Since such a weighting will decrease the impact of errors regarding classes with only a few occurrences, the developed software displays the single class performance as well for better analysis.

3. Experimental design

The experiments are separated into three stages. First, they are performed using the RIWA dataset and, to confirm the results on a different domain, repeated on the Cityscapes dataset. Since Cityscapes consists of many more images with higher resolution, training of all CNNs would take too much computational power to justify a training of all networks on it. Therefore, only the best performing network is used to repeat the three stages. Unfortunately, the Cityscapes team does not provide the testing ground truth as they are calculating and comparing all uploaded testing results with their own implementation. While this is good to make comparisons fair and representative, to conduct a comparison for this research, the same implementation must be used. However, since no tuning of hyperparameters is done, the validation data can be treated as testing dataset. Therefore, the testing *mIoU* refers to the validation *mIoU*. The three stages are setup as follows:

1. In stage 1, all networks are trained on the RIWA dataset **without augmentation** for 150 epochs to create a baseline.
2. Afterwards (stage 2), all networks are trained on the **offline augmented** RIWA dataset for 150 epochs as well.
3. Stage 3 refers to the **online augmentation**. All networks are trained for 200 epochs on the RIWA dataset but with online augmentation and the *geo first* policy.

3.1. Hardware

The experiments are executed partially on the High Performance Computing (HPC) system of the TU Dresden and on a local workstation. When executing a training on the HPC, the following resources are used: 12 Cores @2,3 GHz, 24 GB RAM, NVIDIA A100-SXM4 (40 GB RAM). In total, 256 A100 GPUs are installed on the cluster. The workstation contains the following hardware: AMD Ryzen Threadripper 3990X, 64 Cores @2,9 GHz, 256 GB RAM, 2 *x* NVIDIA RTX A6000 (48 GB RAM). Every training run uses only a single GPU, regardless of the location (HPC or workstation).

Important sidenote: Results of the HPC and the Workstation differ even with exact same settings and active reproducibility, speaking for a device and/or driver dependency or the aforementioned non-deterministic CUDA implementations (Section 2.3). Therefore, when comparing approaches against each other, always the same hardware is used.

3.2. Workflow

Stage 1 refers to the creation of the baselines by using only the initial datasets without any manipulation.

In **stage 2**, the offline augmentation, a n_{total} of 15 is used for both datasets. The goal is to contain many radiometric and geometric augmentations at once, to overcome different lighting situations and different water body colors, caused by different sediments. This approach lets the network focus more on the structure and shapes rather than colors as humans are also able to segment water in grayscale images and under various different conditions. Similar reasons arise for the Cityscapes dataset. A simple example was already shown in Section 4.2: the color of bicycles and bicyclists, cars or sidewalks is not as important as their shapes and structures. To limit the maximum created images, an upper border of 10000 augmentations is set. Thereby, the RIWA training data is increased by around 13.7 times, resulting in 10789 images. The offline augmented Cityscapes training data contains 12975 images, which is an increment of only 5.2 times. As aforementioned, it is impractical to store many more images on the drive. Since the resolution of Cityscapes images is already larger than those of RIWA, the initial storage size is 7.34 GB as shown Table 1. Using an equal increment of 13.7 times would require much more storage and compute power and is therefore impractical. Furthermore, this allows for a vague statement on choosing the increment factor.

The last step refers to the online augmentation, which relates to **stage 3**. This method uses only data from the original datasets. The technique allows for 29 parameters to be initialized differently (the number of maximum augmentations n_{max}, the global probability P_{global} and the 27 probabilities regarding every operation $P_{operation}$). The 27 operations can be found in appendix Table C.11. As this results in a high number of degrees of freedom, an empirical experiment is conducted to obtain proper approximations for each value.

To get an approximation of the $P_{operation}$, every method has to be tested individually. The goal is to find a probability regarding every method that minimizes the validation loss the most (RIWA dataset). During an execution for a method of interest, each image gets augmented ($P_{global} = 100\%$). Further, only the operation of interest is activated during augmentation ($n_{max} = 1$, all other $P_{operations} = 0\%$). Now, 10 training runs are initialized as follows: the probability of the operation of interest rises from 10% to 100% using a 10% increment per run. The result is then compared against the baseline (no augmentation) and the probability with the highest impact is noted. A total of 270 training runs (27 operations à 10 runs) are performed on the HPC. To reduce computational time, a training execution is limited to 50 epochs.

To approximate P_{global} and n_{max}, the following procedure is used: for every$n_{max} = \{1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12\}$, start 10 training runs representing each$P_{global} = \{10; 20; 30; 40; 50; 60; 70; 80; 90; 100\}\%$. Adding up (10 × 12), another 120 training runs are conducted on the HPC of the TU Dresden.

Finally, the best policy is of interest. Using the results of the previous approximation, every policy is tested. Since the determination of approximation values of the probability distributions requires a lot of computation power, it is not practical in many applications. Therefore, also two intuitive runs are added to verify whether this approach is convenient. The parameters are selected based on the thoughts mentioned in Section 3.

Since online augmentation theoretically improves the dataset variance even more than the offline augmentation, finding a local minimum (minimize loss) is expected to take more iterations. Therefore, the number of epochs is set to 200. All executions regarding the policy test are performed on the workstation (Section 3.1).

3.3. Test on Cityscapes

To test the applicability of the developed algorithm for offline and online augmentation on another dataset, we used Cityscapes. Since we are not interested in the best CNN for this dataset, we did not compare all CNNs, as we estimated that this would require more than four additional years (1477 days) of training on a single A100-SMX4 GPU. Therefore we test the approach using only the U-Net.

To obtain the best policy, the probabilities of the online augmentation operations are determined as described previously (Section 3.2). As the dataset is much larger and due to timing restrictions on the HPC, the approximation of n_{max} and P_{global} are not possible. Therefore, the number of augmentations and the global probability are intuitively set to: $n_{max} = 5$ and $P_{global} = 75\%$. Due to an operational error, one training run uses a n_{max} of 8.

The approximation again requires a lot of computational power. Therefore, two tests with intuitive parameters are also performed. All policy tests are executed on the aforementioned workstation and the parameter approximation is done on the HPC (Section 3.1).

4. Results and discussion

Due to the large amount of networks, only the best 5 architectures are shown throughout the paper. The term "best" refers to the averaged *IoU* regarding the three strategies applied on the RIWA dataset $((IoU_{no\ aug} + IoU_{offline\ aug} + IoU_{online\ aug})/3)$. The full size comparison can be found in appendix Table E.15.

Table 1
Sample and storage sizes of the training data regarding the RIWA and Cityscapes dataset.

	RIWA		Cityscapes	
	No Augmentation	Offline Augmentation	No Augmentation	Offline Augmentation
Samples	789	10789	2975	12975
Storage size	512 MB	15.2 GB	7.34 GB	33.0 GB

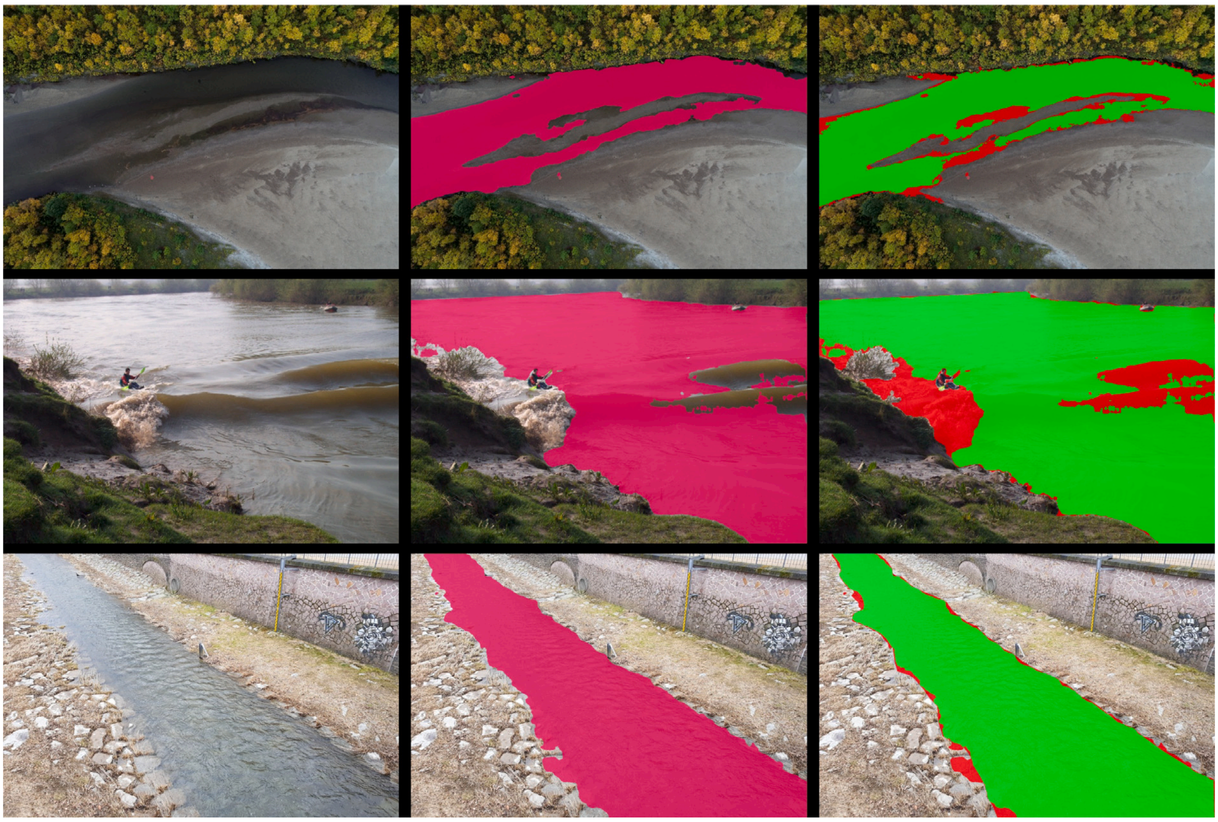


Fig. 7. Example segmentations of the RIWA dataset and the U-Net (ResNeXt 50) without augmentation. Left: Input image; Center: Segmentation as overlay; Right: Difference map between the ground truth and the segmentation (green: segmentation = ground truth, red: segmentation != ground truth).

Table 2
Top 5 performing networks trained on the RIWA dataset without augmentation. *IoU* and Accuracy are related to the test data, Loss to the validation data.

Network	*IoU*	Accuracy (%)	Loss
U-Net (ResNeXt 50)	**0.928**	**97.531**	**0.118**
UPerNet (ResNeXt 50)	0.923	97.357	0.121
GCN (ResNeXt 50)	0.916	97.207	0.130
DANet (ResNeXt 50)	0.908	96.858	0.122
SegNet (ResNeXt 50)	0.906	96.597	0.141

Table 3
Top 5 performing networks trained on the RIWA dataset with offline augmentation. *IoU* and Accuracy are related to the test data, Loss to the validation data.

Network	IoU	Accuracy (%)	Loss
U-Net (ResNeXt 50)	**0.980**	**99.355**	0.032
UPerNet (ResNeXt 50)	0.979	99.345	**0.026**
SegNet (ResNeXt 50)	0.976	99.228	0.034
GCN (ResNeXt 50)	0.975	99.200	0.032
DANet (ResNeXt 50)	0.971	99.082	0.031

4.1. Stage 1: Creating a baseline

As shown in Table 2, the U-Net has proven its known ability to perform well on a small dataset (Ronneberger et al., 2015) and achieves the highest *IoU*, closely followed by the UPerNet and GCN. However, the segmentations in Fig. 7 contain many errors and would result in poor automatic gauge detection.

4.2. Stage 2: Using offline augmentation

Table 3 proves, that the offline augmented dataset significantly increases $IoUs$ compared to the baseline ($\overline{\Delta}_{IoU} = 0.06$). The U-Net has, the highest IoU whereas the UPerNet produces a smaller error on the validation data. The fluctuation in the intersection over union is even smaller than without augmentation. In fact, differences on such a small scale may be caused due to the weight initialization of the networks. A different seed during the initialization would lead to different results. This suggests that the top 5 networks may generalize equally well given a sufficient amount of augmented data. Whether this pattern continues will be seen in the following section.

In Fig. 8 some segmentation examples using the U-Net (ResNeXt 50) trained on the offline augmented dataset are shown. The visual quality is much better than without augmentation.

4.3. Stage 3: Using online augmentation

To compute the parameter approximations, the UPerNet (ResNext 18) is used to allow faster computation. It has been observed, that a $n_{max} = 8$ and a $P_{global} = 40\%$ are advantageous. The probabilities for the 27 operations can be found in the appendix Table C.11. The most useful

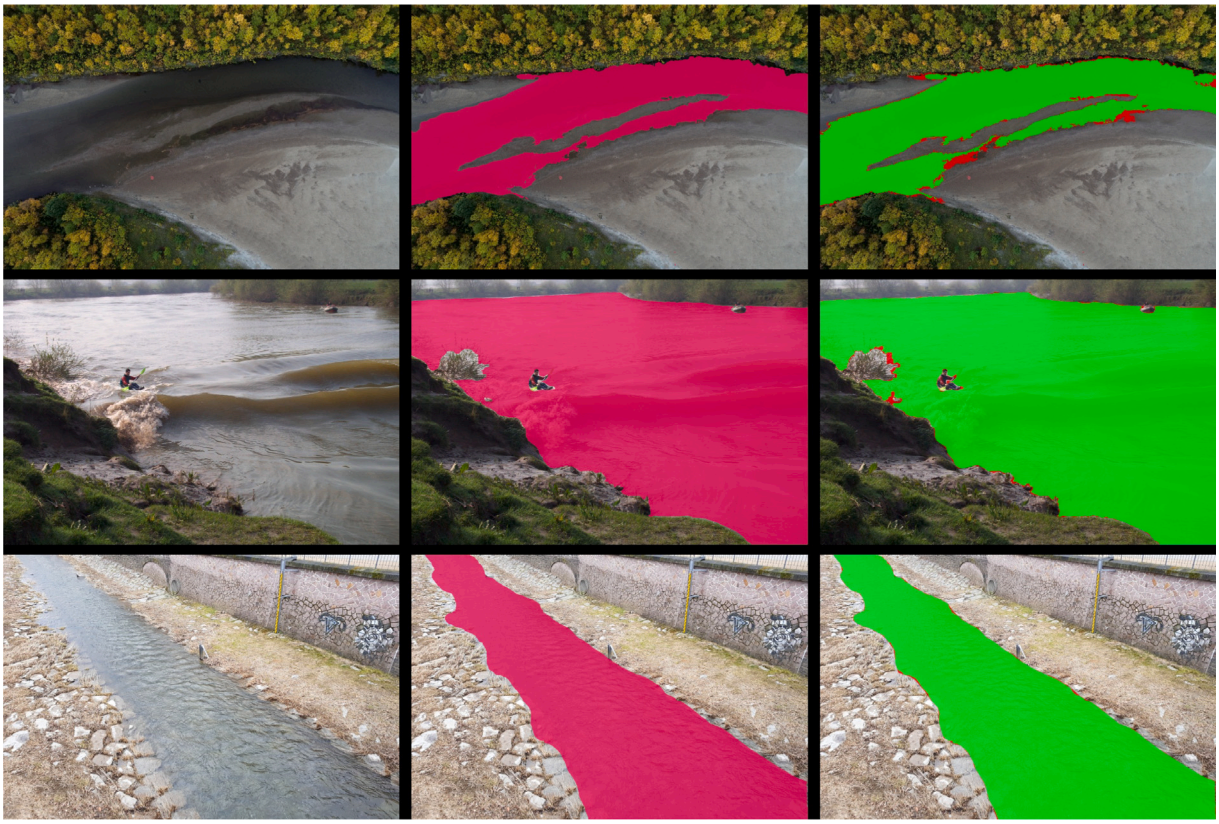


Fig. 8. Examples of a segmentation using the offline augmented RIWA dataset and the U-Net (ResNeXt 50). Left: Input image; Center: Segmentation as overlay; Right: Difference map between the ground truth and the segmentation (green: segmentation = ground truth, red: segmentation != ground truth). More and larger images in the appendix Fig. F.16.

Table 4
The five most useful online augmentation methods with their estimated activation probabilities.

Rank	Method	$P_{operation}$ (%)
1.	Gaussian Noise	100
2.	Squeeze	90
3.	Resize	70
4.	Elastic distortion	70
5.	Rotation	70

Table 5
Online augmentation policy comparison on the RIWA dataset using the U-Net (ResNeXt 50). *IoU* and Accuracy are related to the test data, Loss to the validation data.

	Policy	*IoU*	Accuracy (%)	Loss
	Baseline	0.909	97.02	0.124
1	*Geo first*	0.921	97.27	0.106
2	*Pix first*	0.901	96.64	0.103
3	*Geo only*	**0.927**	**97.45**	0.117
4	*Pix only*	0.911	96.91	0.109
5	*Random*	0.912	96.48	0.108
6	Intuitive values + *geo first*	0.922	97.28	0.096
7	Intuitive values + *random*	0.918	96.75	**0.094**

operations are presented in Table 4. The parameter approximation has shown that Gaussian noise addition is the most beneficial augmentation ($P_{operation}$ = 100%; every image was augmented) followed by random squeezing ($P_{operation}$ = 90%). The average $P_{operation}$ of all geometric augmentations is 53% and for the radiometric augmentations, the average is 32%.

Using those findings, all online augmentation policies are tested. The results are shown in Table 5 (using the U-Net (ResNeXt 50), highest IoU at baseline and offline augmentation). As aforementioned, the parameter computation is not practical and two intuitive runs are added based on the considerations in Section 2.2.2. The intuitive parameters can be found in appendix Table C.12.

The *geo only* policy (3 in Table 5) using the approximated values leads to the best testing *IoU*. It is noteworthy that using an intuitive approach plus the *geo first* policy (6 in Table 5) achieves an *IoU* delta of only −0.005 compared to the *geo only* method, which again can be explained by the weight initialization. The loss regarding the validation data is even smaller.

The intuitive *random* policy (7 in Table 5) performs worse on the test data using the same intuitive values. However, this policy does also not perform well using the approximated values.

The *IoU* of the *pix first* policy is even smaller than without augmentation. One possible reason for this is the use of many different cameras while capturing the training data. In addition, the dataset also covers different lighting situations. This leads to the assumption that the radiometric image space might already be well covered during the day. Therefore, the *geo only* approach should perform well. However, the Δ_{IoU} of the *geo only* policy vs. the baseline (U-Net (ResNeXt 50)) is only increased by 0.18. Comparing the offline augmentation against the baseline (U-Net (ResNeXt 50)), a Δ_{IoU} of 0.52 is obtained. It appears that augmenting the entire image instead of smaller patches, as is the case for online augmentation where the sub-patches are used, is beneficial.

In summary, such an extensive testing procedure is optional if the user has a good intuition in selecting the augmentation parameters. The software is able to generate and visualize random examples representing the current settings to assist the user in setting the correct parameters. This allows for better fine-tuning of the selected operations.

Using the determined parameters (appendix Table C.11) and the policies *geo only* and *geo first*, all 91 networks are trained on the HPC of the TU Dresden. In Table 6, the *geo first* policy is shown, as the results on the HPC are generally better than choosing only geometric augmentations (see Table 9). The reason for this difference seems to

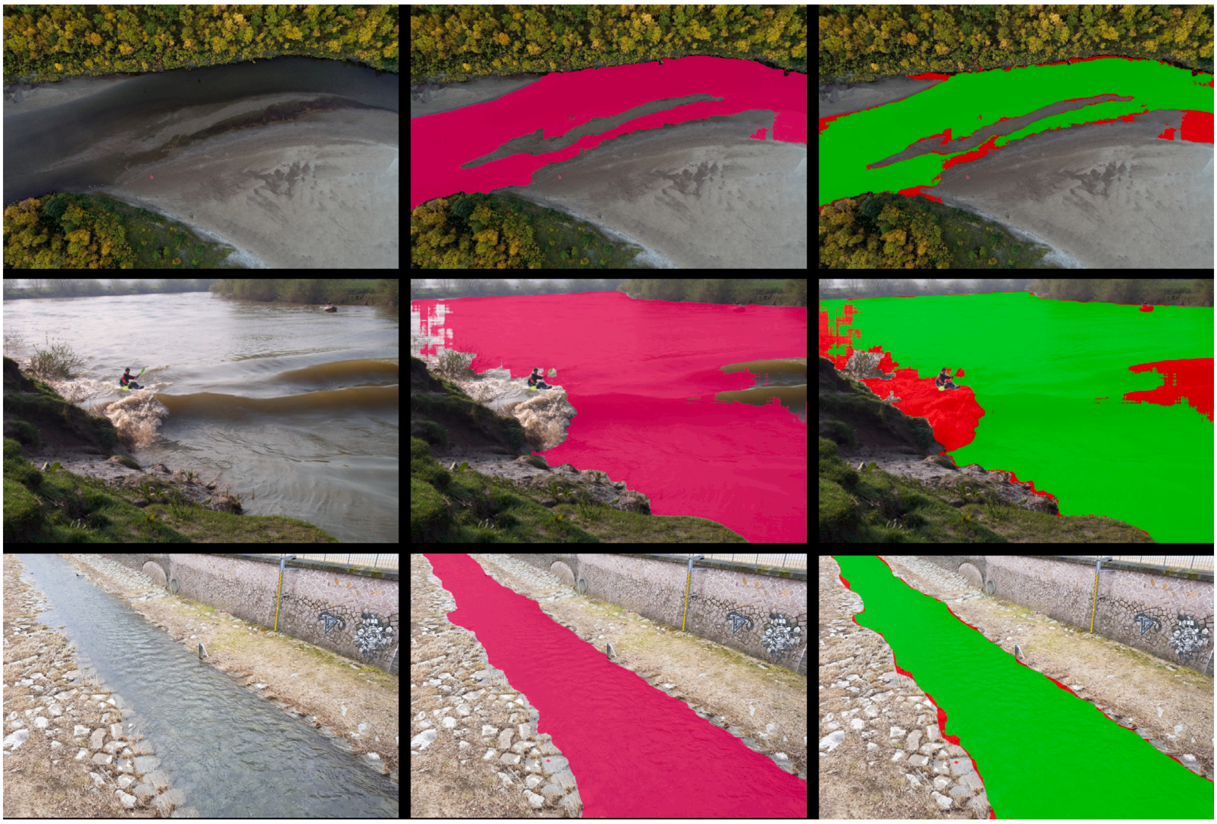


Fig. 9. Example segmentations of the RIWA dataset and the U-Net (ResNeXt 50) trained on the HPC of the TU Dresden with online augmentation (*intuitive* + *geo first*). Left: Input image; Center: Segmentation as overlay; Right: Difference map between the ground truth and the segmentation (green: segmentation = ground truth, red: segmentation != ground truth).

Table 6
Top 5 performing networks trained on the RIWA dataset with online augmentation (policy *geo first*). *IoU* and Accuracy are related to the test data, Loss to the validation data.

Network	IoU	Accuracy (%)	Loss
GCN (ResNeXt 50)	**0.924**	97.136	0.102
U-Net (ResNeXt 50)	0.922	**97.289**	0.103
UPerNet (ResNeXt 50)	0.921	96.973	**0.099**
SegNet (ResNeXt 50)	0.919	97.199	0.102
DANet (ResNeXt 50)	0.918	97.195	0.106

be the varying hardware and the non-determinism of some CUDA implementations (Section 3.1). To find out the exact cause, further comprehensive tests would have to be carried out, which is not done in this research.

The fluctuation in Table 6 is again very small. As already mentioned in Section 4.2, this can be explained due to the initialization of the networks and here also due to the randomness of the online augmentation. The observed pattern that the top 5 networks generalize similarly well when the dataset variance is high becomes apparent again (Section 4.2). However, the visual quality of the exemplary segmentations as shown in Fig. 9 are not sufficient and seem to be even worse than without augmentation.

4.4. Test on Cityscapes

To test the algorithms in a different domain, the Cityscapes dataset is used. The approximated and intuitive parameters are shown in Appendix D. The parameter approximation has shown that random erasing has the highest $P_{operation}$ with 100% followed by random squeezing ($P_{operation} = 90\%$). The average $P_{operation}$ of all geometric augmentations is 37% and for the radiometric augmentations, the average is 43%.

Table 7
Online augmentation policy comparison on the Cityscapes dataset using the U-Net (ResNeXt 50). *mIoU*, Accuracy and Loss are related to the validation data.

	Policy	*mIoU*	Accuracy (%)	Loss
	Baseline	0.861	90.466	0.379
1	*Geo first*	0.867	91.093	0.325
2	*Pix first*	0.870	91.288	0.324
3	*Geo only*	0.862	90.554	0.345
4	*Pix only*	0.863	90.860	0.333
5	*Random*	0.862	90.824	0.327
6	*Geo first* $n_{max} = 8$	**0.872**	**91.430**	0.326
7	Intuitive values + *geo first*	0.863	90.804	0.325
8	Intuitive values + *random*	0.865	90.947	**0.314**

In Table 7, the results using the different policies are shown. The increases in the *mIoU* are smaller compared to the RIWA dataset because this dataset is larger and may have higher initial variance. However, every policy leads to a small increment of the *mIoU*. The best result is obtained with the approximated parameters and a higher n_{max} of 8. The *pix first* policy also achieves a high *mIoU*. This is probably due to the fact, that it is the shape of cars, bikes, facades, etc. that are of interest, not the color space. By manipulating the color space in almost every augmented image, the network focuses on these shapes rather than the colors.

In contrast to the small RIWA dataset, the offline augmentation did not outperform the online augmentation (Table 8) due to overfitting (see Figs. 11 and 12). This can be explained by the fact that a smaller dataset was created compared to the approach used to augment the RIWA dataset (5.2 times using Cityscapes, 13.7 times using RIWA). Therefore we also conducted an experiment, increasing the Cityscapes dataset by 13.7 times, which interestingly did not improve the performance nor prevented overfitting. However, the required storage space

Table 8
Comparison of all training strategies regarding the Cityscapes and RIWA dataset using the U-Net (ResNeXt 50).

U-Net (ResNext 50)	Cityscapes (validation data)			RIWA		
Augmentation	None	Offline	Online (*geo first*)	None	Offline	Online (*geo only*)
(m-) IoU	0.861	0.860	0.872	0.909	0.978	0.927
Accuracy (%)	90.466	90.579	91.430	97.018	99.299	97.454
Best epoch	17	6	66	16	67	104
Duration per epoch	18 m 07 s	87 m 19 s	18 m 11 s	4 m 04 s	39 m 13 s	4 m 20 s
Training images	2975	12975	2975	789	10789	789

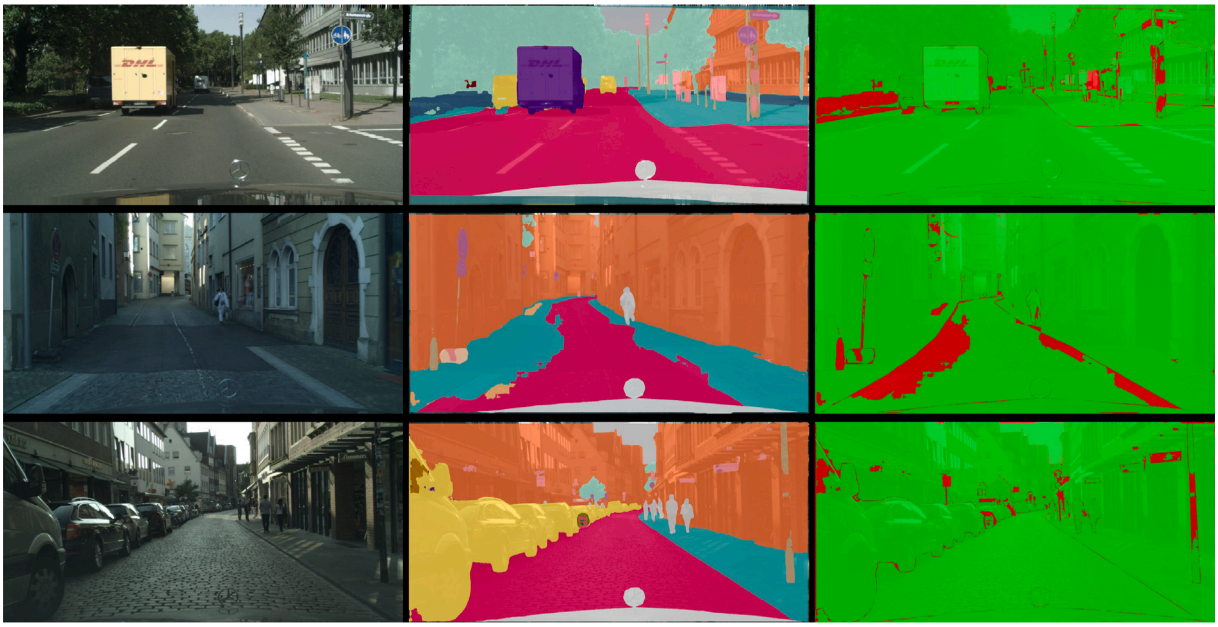


Fig. 10. Examples of a segmentation using the Cityscapes dataset with online augmentation and the U-Net (ResNeXt 50). Left: Input image; Center: Segmentation as overlay; Right: Difference map between the ground truth and the segmentation (green: segmentation = ground truth, red: segmentation != ground truth). More and larger images in the appendix Fig. F.15.

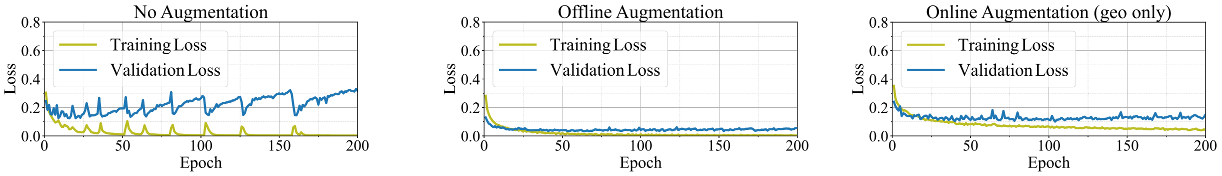


Fig. 11. Training and validation loss on the RIWA dataset using the different training strategies. The U-Net (ResNeXt 50) was used as CNN.

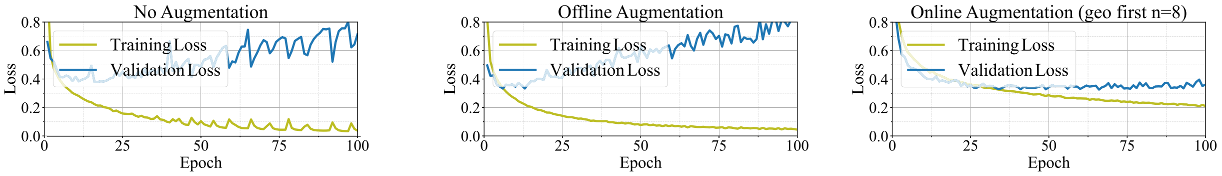


Fig. 12. Training and validation loss on the Cityscapes dataset using the different training strategies. The U-Net (ResNeXt 50) was used as CNN.

was increased to 80 GB, which is not practical to store considering both space requirements and training duration. Therefore, the online augmentation is beneficial for larger datasets and may be improved in future studies to augment whole images instead of patches.

In order to achieve the best possible results with the Cityscapes dataset, further testing must be performed. Since the purpose of this benchmark is not to understand the urban scene, but to determine which augmentation method is more beneficial, no further tests, i.e. training on a large offline augmented version, are performed. In Fig. 10 some segmentation examples are shown using the online augmentation (*geo first*) approach and the U-Net (ResNeXt 50).

4.5. Future work - a new online augmentation

As described in Section 4.3, a next step is to implement an online augmentation procedure in which the entire image is augmented instead of a partial image. However, this introduces a new imbalance problem. After the online augmentation, a sub-region of the augmented image is passed to the network. Consider a network with an input size of 512 × 512. If a training sample is of the same size, the same section will be shown to the network at each epoch. Selecting a random sub-image from 1536 × 1536 resolution image would most likely never

Table 9
Direct comparison of 32 CNNs trained on the RIWA dataset. Avg *IoU*: averaged testing *IoU* of every strategy; No Augmentation: testing *IoU* without augmentation; Offline: testing *IoU* on the offline augmented dataset; Online: testing *IoU* using online augmentation and the policies *geo first* and *geo only*.

	Network	Batch size	Avg *IoU*	No Aug	Offline	Online (*geo first*)	Online (*geo only*)
1.	U-Net (ResNeXt 50)	24	**0.940**	**0.928**	**0.980**	0.928	0.922
2.	GCN (ResNeXt 50)	48	0.935	0.916	0.975	0.925	**0.924**
3.	UPerNet (ResNeXt 50)	32	0.935	0.923	0.979	0.919	0.921
4.	SegNet (ResNeXt 50)	24	0.932	0.906	0.976	0.928	0.919
5.	DANet (ResNeXt 50)	32	0.931	0.908	0.971	**0.929**	0.918
6.	Dran (ResNeXt 50)	40	0.929	0.903	0.969	0.924	0.922
7.	OCNet (ResNeXt 101)	22	0.929	0.904	0.970	0.928	0.915
8.	PSPNet (DenseNet 121)	16	0.929	0.901	0.973	0.928	0.915
9.	DenseASPP 121	18	0.923	0.884	0.972	0.920	0.915
10.	ICNet (ResNeXt 101)	64	0.923	0.890	0.969	0.922	0.910
Comparison of all 32 networks in the appendix (Table E.15)							

result in exactly the same sub-image, which would probably result in overfitting of the network to the smaller images.

5. Conclusions

To summarize the CNN comparison on the RIWA dataset, a direct comparison of the three strategies using all 32 CNNs is shown in Table 9. In terms of online augmentation, geometric operations benefited the most, with squeezing and resizing, being the top two geometric augmentation operations. In the radiometric space, Gaussian noise and brightness change were the most beneficial methods. In terms of augmentation strategy, offline augmentation performed significantly better than online augmentation:

- Mean improvement using offline augmentation: $\overline{\Delta}_{IoU} = 0.083$.
- Mean improvement using online augmentation (intuitive *geo first*): $\overline{\Delta}_{IoU} = 0.035$.
- Mean improvement using online augmentation (*geo only*): $\overline{\Delta}_{IoU} = 0.017$.

A likely reason for the superiority of the offline augmentation is the input image of the augmenters. During offline augmentation, the entire image is augmented, whereas in online augmentation, only a sub region of the image is augmented.

It has further been shown that a random concatenation of augmentation methods is less advantageous than using a chaining of first geometric augmentations and afterwards radiometric manipulations.

We also tested the approach on a different domain (Cityscapes) to verify if the algorithms were there useful as well. In contrast to the previous findings, offline augmentation could not increase the IoU nor prevent overfitting. Only the online augmentation was able to stabilize the training and prevent overfitting. Also, more data significantly increased the training time. In fact, Cityscapes constitutes a rather small dataset in comparison to other, well-established datasets such as the COCO dataset (≈118k training images). Hence, online augmentation is the practical approach in such a scenario. Therefore, we present the following recommendation for choosing the augmentation strategy: On very small datasets, which are common in environmental science, offline augmentation may be used. On datasets where the initial image variance is expected to be higher, such as in Cityscapes, online augmentation is superior due to the more versatile chaining of operations.

During all tests that were carried out, it became clear that the U-Net using the ResNeXt 50 backbone was the best CNN for river water segmentation. In addition, the ResNeXt 50 is the best performing feature extractor. It also became clear that all trained networks supporting a backbone were superior to those without a pre-trained encoder.

CRediT authorship contribution statement

Franz Wagner: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Anette Eltner:** Data curation, Writing – review & editing, Supervision, Funding acquisition. **Hans-Gerd Maas:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The link to the published data and used software are given in the paper.

Acknowledgments

The research work presented in the publication has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB/TRR 280; Projekt-ID: 417002380

The research has further been funded by the BMBF (Federal Ministry of Education and Research; ID: 13N15542). We would like to thank the KIWA (Künstliche Intelligenz für die Hochwasserwarnung) project group for their support.

The authors are grateful to the Center for Information Services and High Performance Computing [Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)] at TU Dresden for providing its facilities for high throughput calculations.

Appendix A

See Table A.10.

Appendix B. Augmentation methods

In the following, all possible augmentations are shown. In the text and software, some of them are combined like: tilt backwards, tilt forwards, tilt left, tilt right to: random tilt.

Appendix C. Used RIWA online augmentation parameters

See Tables C.11 and C.12.

Appendix D. Used Cityscapes online augmentation parameters

See Tables D.13 and D.14.

Table A.10
List of all neural networks.

AttentionUNet (Oktay et al., 2018)	BiSeNet (Yu et al., 2018a)	BiSeNet V2 (Yu et al., 2020)	DANet (Fu et al., 2018)
DFANet (Li et al., 2019)	DFN (Yu et al., 2018b)	DeepLab-DUC-HDC (Wang et al., 2017)	DeepLabV3+ (Chen et al., 2018)
DenseASPP (Yang et al., 2018)	DenseNet (Huang et al., 2016)	DRAN (Lyu et al., 2021)	ENet (Paszke et al., 2016)
ERFNet (Romera et al., 2017)	ESPNet (Mehta et al., 2018)	ExtremeC3Net (Park et al., 2019a)	FCN (Shelhamer et al., 2016)
FastSCNN (Poudel et al., 2019)	GCN (Peng et al., 2017)	HRNet (Wang et al., 2019a)	ICNet (Zhao et al., 2017)
LEDNet (Wang et al., 2019b)	LadderNet (Zhuang, 2018)	OCNet (Yuan and Wang, 2018)	PSPNet (Zhao et al., 2016)
R2 Attention U-Net (Alom et al., 2018)	R2U-Net (Alom et al., 2018)	SINet (Park et al., 2019b)	SegNet (Badrinarayanan et al., 2015)
U-Net (Ronneberger et al., 2015)	U-Net ++ (Zhou et al., 2018)	U-Net3+ (Huang et al., 2020)	UPerNet (Xiao et al., 2018)

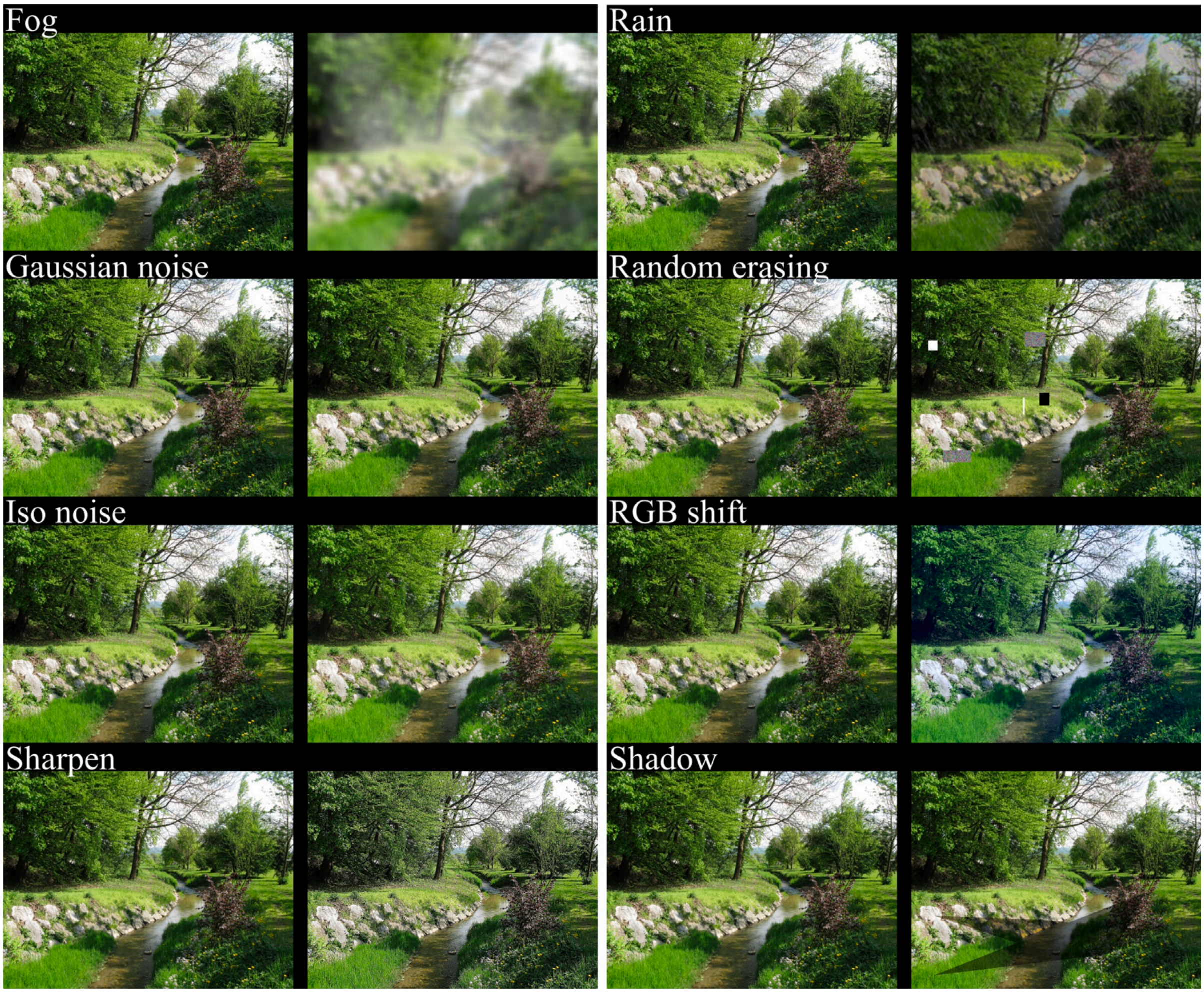


Fig. B.13. Start of example augmentations. Continuing on next 4 pages until Fig. B.14.

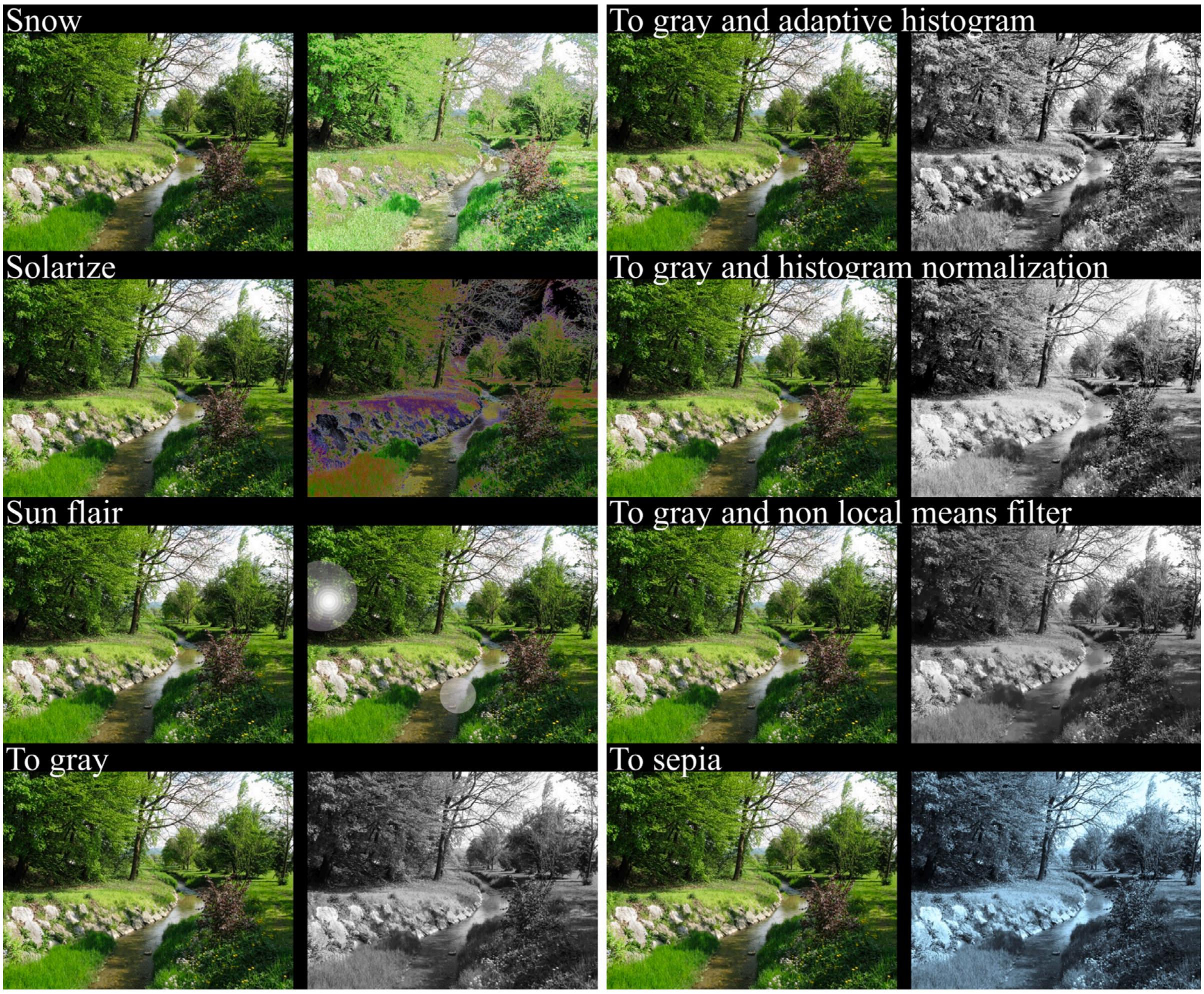

Snow
To gray and adaptive histogram
Solarize
To gray and histogram normalization
Sun flair
To gray and non local means filter
To gray
To sepia

Fig. B.13. (*continued*).

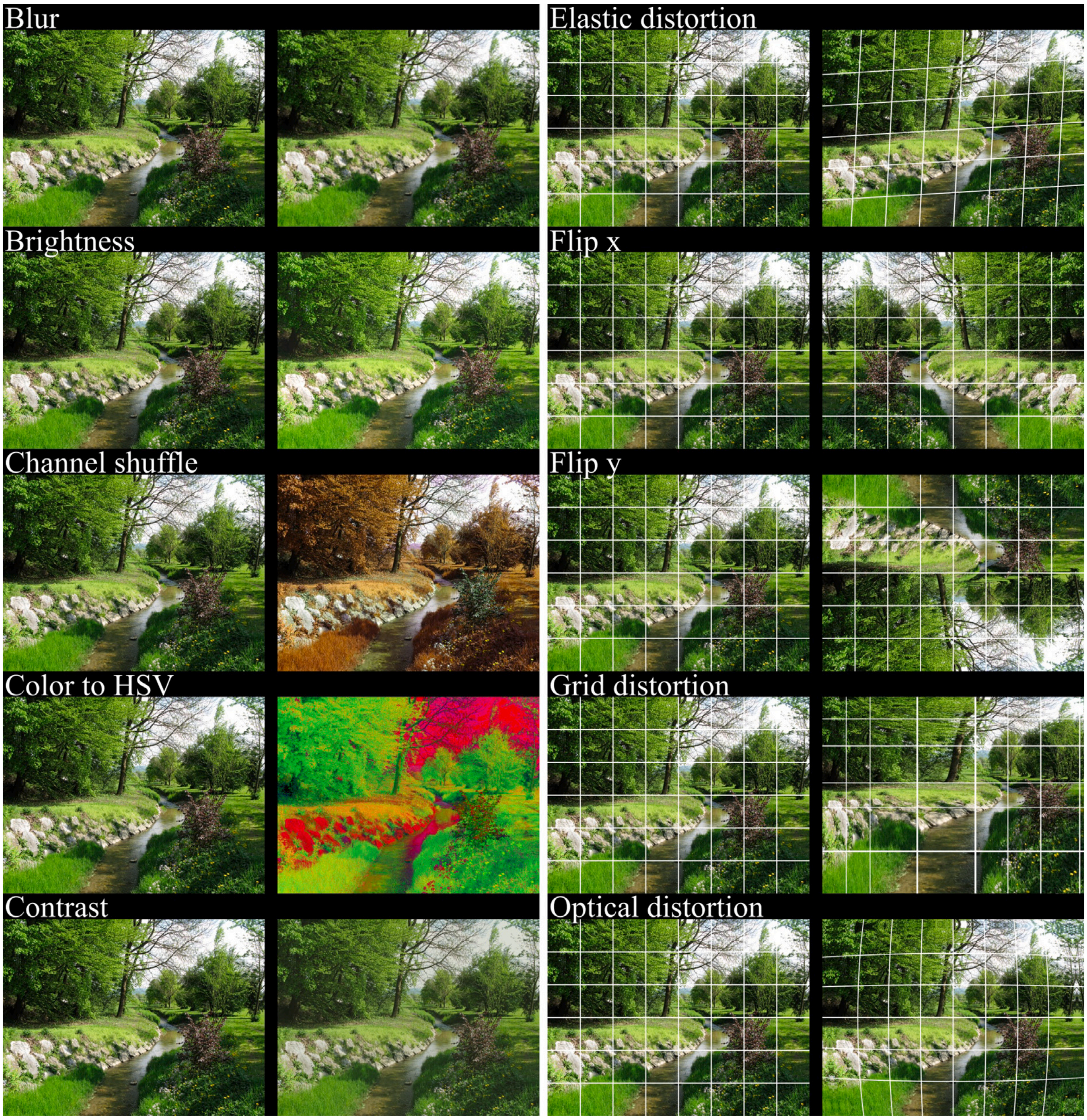


Fig. B.13. (*continued*).

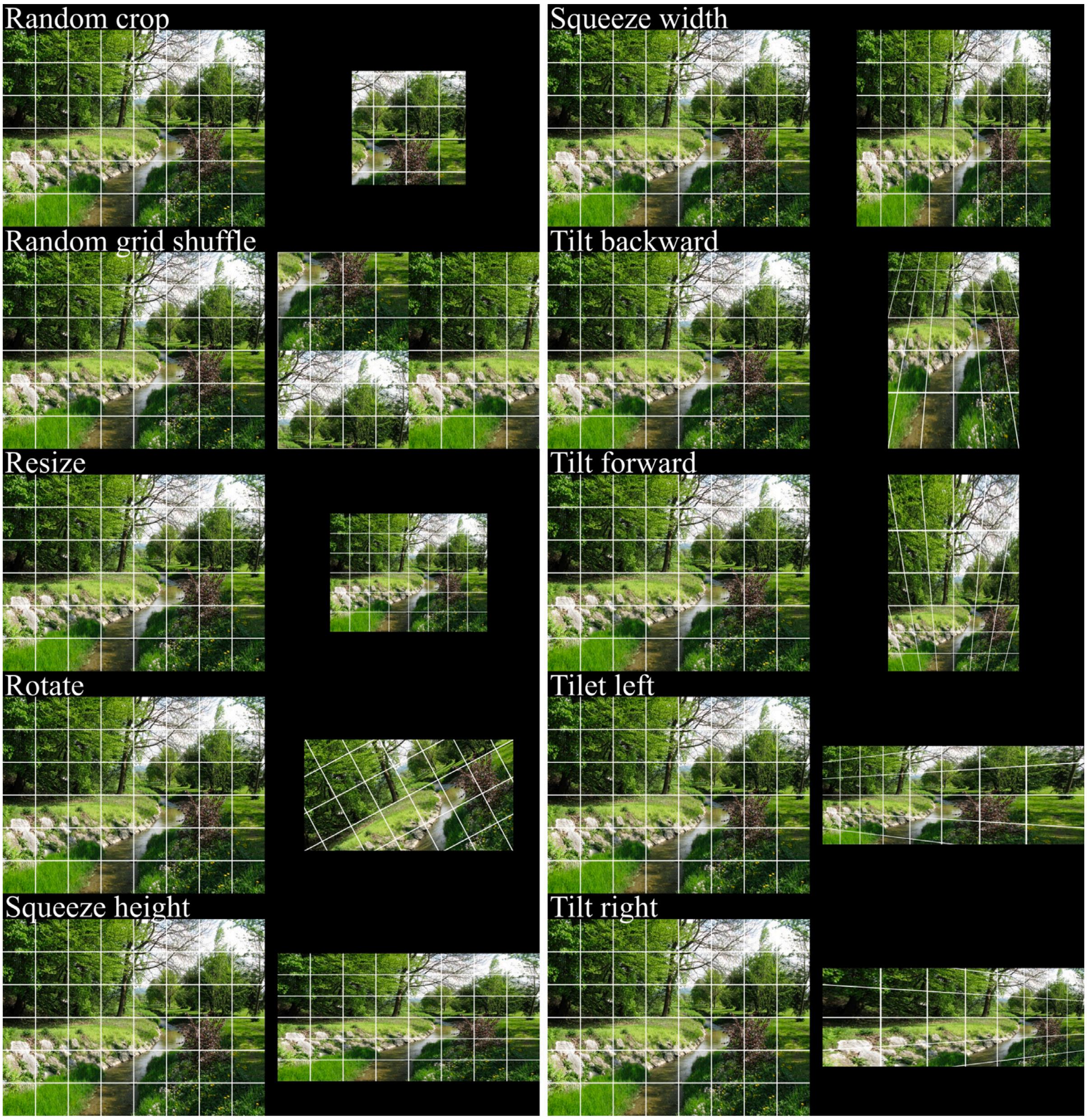


Fig. B.14. All 36 augmentations and related examples. This figure ends all augmentation examples starting from Fig. B.13.

Table C.11
Parameters and approximated activation probabilities of the online augmentation regarding RIWA.

Parameter	Value	Operation	Value
Global strength	40%	Max augmentations	8
Max rotation angle	±30°	Vertical flip allowed	No
Elastic distortion	70%	Flip	60%
Grid distortion	50%	Grid shuffle	0%
Optical distortion	30%	Random crop	50%
Resize	70%	Rotate by angle	70%
Squeeze	90%	Tilt	40%
To sepia	10%	Brightness	60%
Channel shuffle	10%	Color to hsv	20%
Contrast	40%	Gaussian filter	20%
Gaussian noise	100%	Iso noise	20%
Random erasing	50%	Random fog	20%
Random rain	20%	Random shadow	20%
Random snow	40%	Random sun flair	50%
Rgb shift	10%	Solarize	40%
To gray	10%		

Table C.12
Parameters and intuitive activation probabilities of the online augmentation regarding RIWA.

Parameter	Value	Operation	Value
Global strength	75%	Max augmentations	6
Max rotation angle	±30°	Vertical flip allowed	No
Elastic distortion	75%	Flip	50%
Grid distortion	50%	Grid shuffle	0%
Optical distortion	75%	Random crop	0%
Resize	50%	Rotate by angle	75%
Squeeze	33%	Tilt	50%
To sepia	0%	Brightness	50%
Channel shuffle	25%	Color to hsv	0%
Contrast	50%	Gaussian filter	25%
Gaussian noise	50%	Iso noise	50%
Random erasing	50%	Random fog	25%
Random rain	25%	Random shadow	50%
Random snow	50%	Random sun flair	50%
Rgb shift	0%	Solarize	50%
To gray	0%		

Table D.13
Parameters and approximated activation probabilities of the online augmentation regarding Cityscapes.

Parameter	Value	Operation	Value
Global strength	75%	Max augmentations	5
Max rotation angle	±5°	Vertical flip allowed	No
Elastic distortion	10%	Flip	20%
Grid distortion	60%	Grid shuffle	0%
Optical distortion	80%	Random crop	20%
Resize	30%	Rotate by angle	20%
Squeeze	90%	Tilt	40%
To sepia	20%	Brightness	50%
Channel shuffle	70%	Color to hsv	20%
Contrast	60%	Gaussian filter	50%
Gaussian noise	50%	Iso noise	10%
Random erasing	100%	Random fog	30%
Random rain	10%	Random shadow	60%
Random snow	20%	Random sun flair	80%
Rgb shift	50%	Solarize	20%
To gray	30%		

Table D.14
Parameters and intuitive activation probabilities of the online augmentation regarding Cityscapes.

Parameter	Value	Operation	Value
Global strength	75%	Max augmentations	8
Max rotation angle	±5°	Vertical flip allowed	No
Elastic distortion	70%	Flip	50%
Grid distortion	0%	Grid shuffle	0%
Optical distortion	50%	Random crop	0%
Resize	50%	Rotate by angle	75%
Squeeze	25%	Tilt	25%
To sepia	0%	Brightness	50%
Channel shuffle	25%	Color to hsv	25%
Contrast	50%	Gaussian filter	50%
Gaussian noise	50%	Iso noise	50%
Random erasing	50%	Random fog	25%
Random rain	25%	Random shadow	50%
Random snow	25%	Random sun flair	50%
Rgb shift	25%	Solarize	50%
To gray	0%		

Appendix E. Comparison of CNNs with best backbones on RIWA

See Table E.15.

Appendix F. Segmentation results

See Figs. F.15 and F.16.

Table E.15
Direct comparison of 32 CNNs trained on the RIWA dataset. If multiple backbones are supported, only the best performing is shown. A complete comparison with all backbones can be found on the project page.

	Network	Batch size	Testing *IoU* of the augmentation strategies:				
			Avg	No Aug	Offline	Online	
						geo first	*geo only*
1.	U-Net (ResNeXt 50)	24	**0.940**	**0.928**	**0.980**	0.928	0.922
2.	GCN (ResNeXt 50)	48	0.935	0.916	0.975	0.925	**0.924**
3.	UPerNet (ResNeXt 50)	32	0.935	0.923	0.979	0.919	0.921
4.	SegNet (ResNeXt 50)	24	0.932	0.906	0.976	0.928	0.919
5.	DANet (ResNeXt 50)	32	0.931	0.908	0.971	**0.929**	0.918
6.	Dran (ResNeXt 50)	40	0.929	0.903	0.969	0.924	0.922
7.	OCNet (ResNeXt 101)	22	0.929	0.904	0.970	0.928	0.915
8.	PSPNet (DenseNet 121)	16	0.929	0.901	0.973	0.928	0.915
9.	DenseASPP 121	18	0.923	0.884	0.972	0.920	0.915
10.	ICNet (ResNeXt 101)	64	0.923	0.890	0.969	0.922	0.910
11.	U-Net 3+ (ResNet 34)	12	0.919	0.904	0.972	0.908	0.893
12.	BiSeNet	128	0.918	0.889	0.973	0.915	0.894
13.	DeepLab V3+ (ResNet 101)	32	0.918	0.881	0.970	0.926	0.895
14.	ERFNet	56	0.915	0.887	0.957	0.911	0.905
15.	R2 Attention U-Net	5	0.915	0.891	0.970	0.898	0.901
16.	Attention U-Net	14	0.913	0.873	0.965	0.914	0.900
17.	R2U-Net	6	0.913	0.886	0.975	0.907	0.884
18.	DenseNet 103	4	0.912	0.885	0.958	0.911	0.895
19.	ENet	64	0.911	0.886	0.956	0.906	0.898
20.	HRNet	7	0.908	0.866	0.972	0.912	0.882
21.	U-Net 2+	14	0.908	0.868	0.963	0.900	0.900
22.	DeepLab-DUC-HDC	14	0.904	0.896	0.945	0.886	0.889
23.	DFN	24	0.900	0.875	0.952	0.908	0.867
24.	ESPNet	128	0.898	0.871	0.938	0.893	0.890
25.	BiSeNetV2	96	0.895	0.819	0.956	0.914	0.892
26.	SINet	128	0.885	0.844	0.930	0.888	0.880
27.	ExtremeC3Net	64	0.880	0.851	0.918	0.888	0.863
28.	DFANet	64	0.870	0.784	0.951	0.886	0.858
29.	FastSCNN	128	0.867	0.770	0.930	0.890	0.876
30.	LadderNet	40	0.854	0.811	0.909	0.869	0.827
31.	FCN 8	32	0.850	0.816	0.876	0.897	0.810
32.	LEDNet	64	0.841	0.828	0.893	0.826	0.815

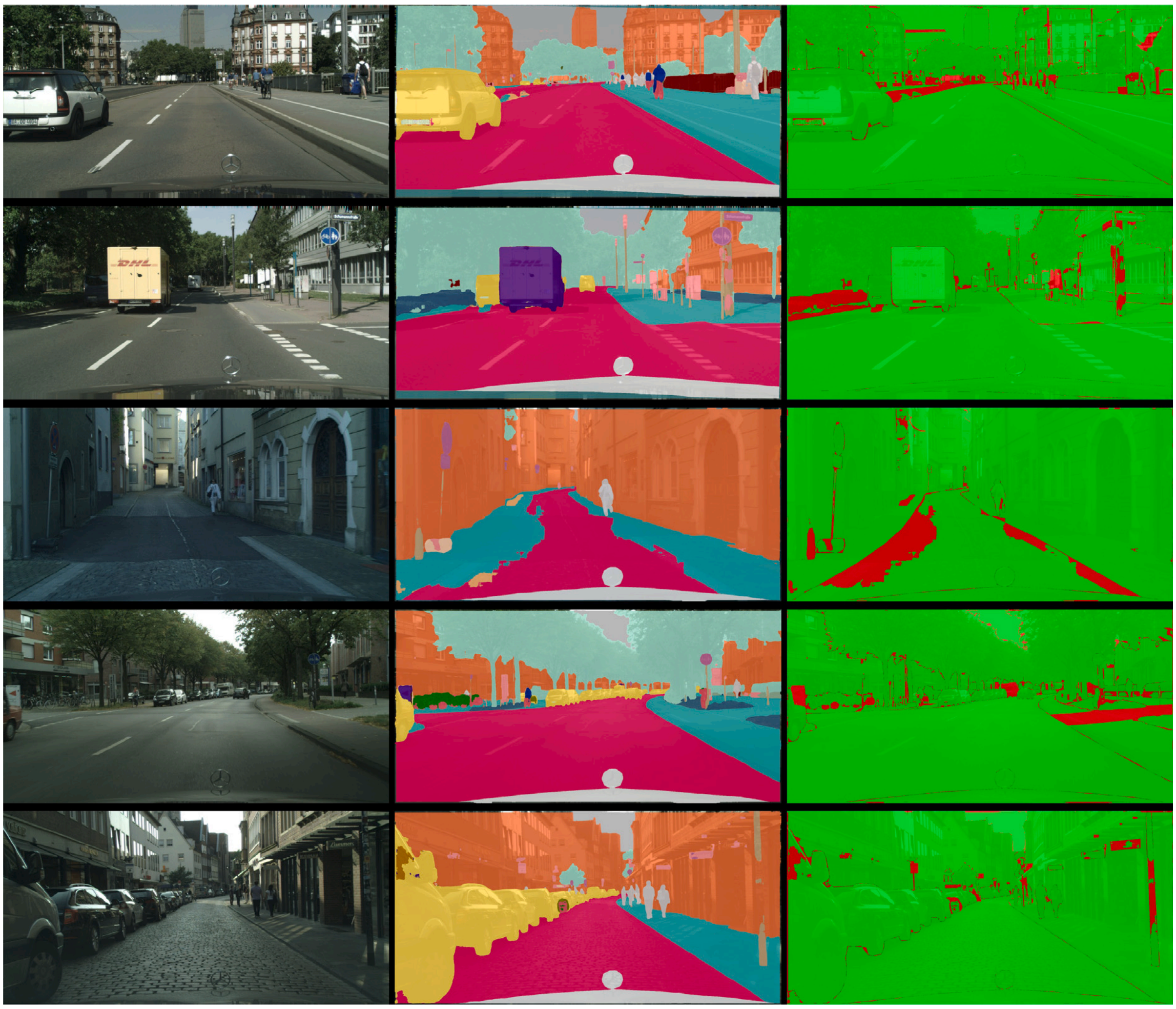


Fig. F.15. Results of a segmentation using the offline augmented Cityscapes dataset and the U-Net (ResNeXt 50). Left: Input image; Center: Segmentation as overlay; Right: Difference map between the ground truth and the segmentation (green: segmentation = ground truht, red: segmentation != ground truth).

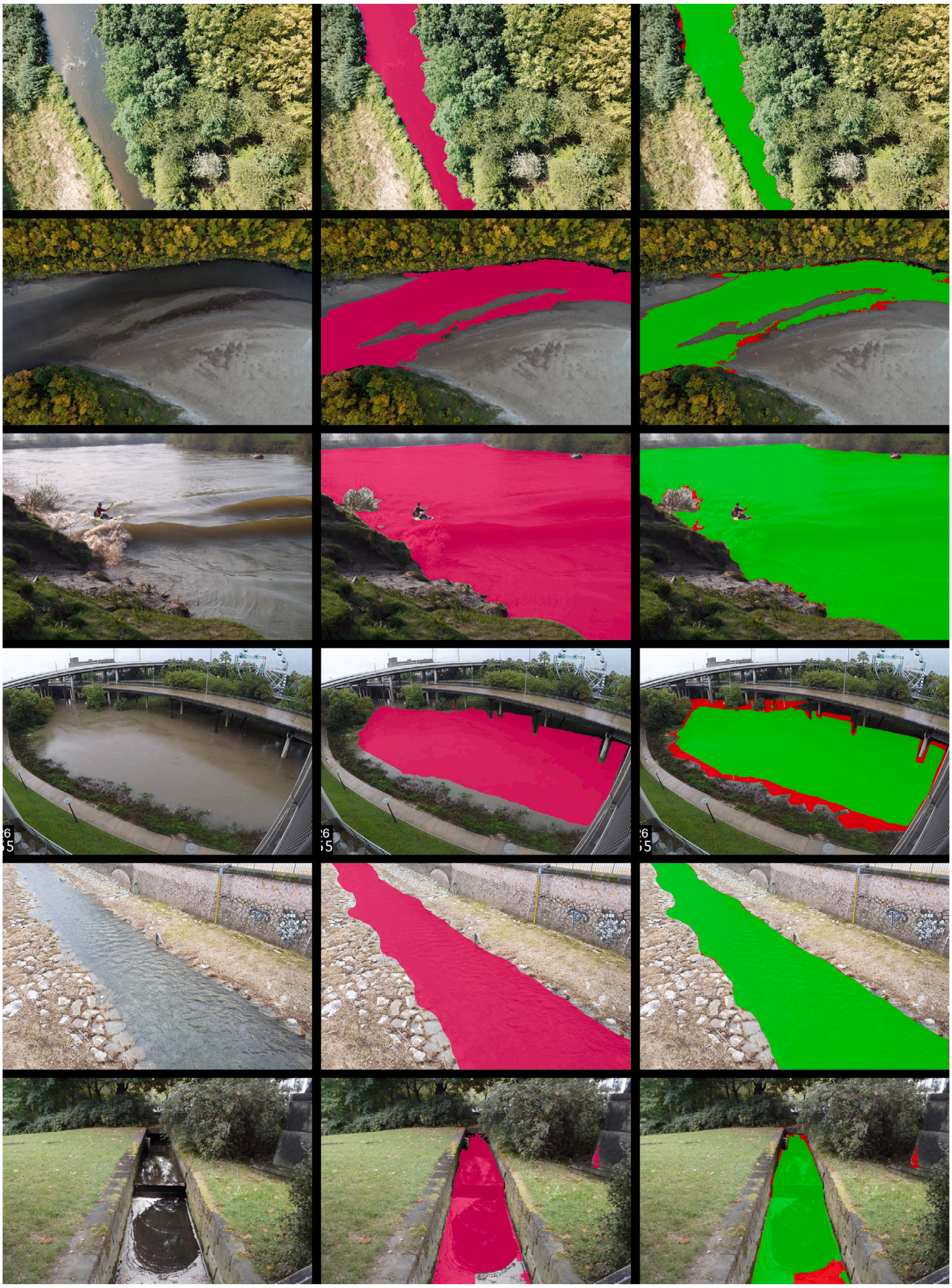


Fig. F.16. Results of a segmentation using the offline augmented RIWA dataset and the U-Net (ResNeXt 50). Left: Input image; Center: Segmentation as overlay; Right: Difference map between the ground truth and the segmentation (green: segmentation = ground truht, red: segmentation != ground truth).

References

Alam, M., Wang, J.-F., Guangpei, C., Yunrong, L.V., Chen, Y., 2021. Convolutional neural network for the semantic segmentation of remote sensing images. Mob. Netw. Appl. 26 (1), 200–215. http://dx.doi.org/10.1007/s11036-020-01703-3.

Alom, M.Z., Hasan, M., Yakopcic, C., Taha, T.M., Asari, V.K., 2018. Recurrent residual convolutional neural network based on U-Net (R2U-Net) for medical image segmentation. http://dx.doi.org/10.48550/arXiv.1802.06955, CoRR abs/1802.06955.

Badrinarayanan, V., Handa, A., Cipolla, R., 2015. SegNet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. http://dx.doi.org/10.48550/arXiv.1505.07293, CoRR abs/1505.07293.

Blanch, X., Wagner, F., Eltner, A., 2022a. RIWA Dataset. Kaggle, http://dx.doi.org/10.34740/KAGGLE/DSV/4289421.

Blanch, X., Wagner, F., Hedel, R., Grundmann, J., Eltner, A., 2022b. Towards automatic real-time water level estimation using surveillance cameras. EGU22 http://dx.doi.org/10.5194/egusphere-egu22-3225.

Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A., 2020. Albumentations: Fast and flexible image augmentations. Information 11 (2), 125. http://dx.doi.org/10.3390/info11020125.

Chen, L., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. http://dx.doi.org/10.48550/arXiv.1802.02611, CoRR abs/1802.02611.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR, http://dx.doi.org/10.48550/arXiv.1604.01685.

Cubuk, E.D., Zoph, B., Mané, D., Vasudevan, V., Le, Q.V., 2019a. AutoAugment: Learning augmentation strategies from data. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 113–123. http://dx.doi.org/10.1109/CVPR.2019.00020.

Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V., 2019b. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. arXiv, http://dx.doi.org/10.48550/ARXIV.1909.13719.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. Ieee, pp. 248–255. http://dx.doi.org/10.1109/CVPR.2009.5206848.

Elias, M., Kehl, C., Schneider, D., 2019. Photogrammetric water level determination using smartphone technology. Photogramm. Rec. 34 (166), 198–223. http://dx.doi.org/10.1111/phor.12280.

Elias, M., Maas, H.-G., 2022. Measuring water levels by handheld smartphones – A contribution to exploit crowdsourcing in the spatio temporal densification of water gauging networks. Int. Hydrogr. Rev. 27, 9–22. http://dx.doi.org/10.58440/ihr-27-a01.

Eltner, A., Bressan, P., Akiyama, T., Gonçalves, W., Junior, J., 2021. Using deep learning for automatic water stage measurements. Water Resour. Res. 57, http://dx.doi.org/10.1029/2020WR027608.

Fu, J., Liu, J., Tian, H., Fang, Z., Lu, H., 2018. Dual attention network for scene segmentation. http://dx.doi.org/10.48550/arXiv.1809.02983, CoRR abs/1809.02983.

Gatys, L.A., Ecker, A.S., Bethge, M., 2015. A Neural Algorithm of Artistic Style. arXiv, http://dx.doi.org/10.48550/ARXIV.1508.06576.

Gong, C., Wang, D., Li, M., Chandra, V., Liu, Q., 2020. KeepAugment: A Simple Information-Preserving Data Augmentation Approach. arXiv, http://dx.doi.org/10.48550/ARXIV.2011.11778.

Huang, H., Lin, L., Tong, R., Hu, H., Qiaowei, Z., Iwamoto, Y., Han, X.-H., Chen, Y.-W., Wu, J., 2020. UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation. pp. 1055–1059. http://dx.doi.org/10.1109/ICASSP40776.2020.9053405.

Huang, G., Liu, Z., Weinberger, K.Q., 2016. Densely connected convolutional networks. http://dx.doi.org/10.48550/arXiv.1608.06993, CoRR abs/1608.06993.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. http://dx.doi.org/10.48550/arXiv.1412.6980.

Li, H., Xiong, P., Fan, H., Sun, J., 2019. DFANet: Deep feature aggregation for real-time semantic segmentation. http://dx.doi.org/10.48550/arXiv.1904.02216, CoRR abs/1904.02216.

Liang, Y., Jafari, N., Luo, X., Chen, Q., Cao, Y., Li, X., 2020. Kaggle WaterNet: An adaptive matching pipeline for segmenting water with volatile appearance. Comput. Vis. Media 1–14. http://dx.doi.org/10.1007/s41095-020-0156-x.

Lyu, Y., Chen, P., Sun, J., Wang, X., Dong, J., Tan, T., 2021. DRAN: Detailed Region-Adaptive Normalization for Conditional Image Synthesis. arXiv, http://dx.doi.org/10.48550/ARXIV.2109.14525.

Mehta, S., Rastegari, M., Caspi, A., Shapiro, L.G., Hajishirzi, H., 2018. ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. http://dx.doi.org/10.48550/arXiv.1803.06815, CoRR abs/1803.06815.

Morin, M., Willetts, M., 2020. Non-determinism in TensorFlow ResNets. http://dx.doi.org/10.48550/arXiv.2001.11396, CoRR abs/2001.11396.

Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M.C.H., Heinrich, M.P., Misawa, K., Mori, K., McDonagh, S.G., Hammerla, N.Y., Kainz, B., Glocker, B., Rueckert, D., 2018. Attention U-Net: Learning where to look for the pancreas. http://dx.doi.org/10.48550/arXiv.1804.03999, CoRR abs/1804.03999.

Park, H., Sjösund, L.L., Yoo, Y., Kwak, N., 2019a. ExtremeC3Net: Extreme lightweight portrait segmentation networks using advanced C3-modules. http://dx.doi.org/10.48550/arXiv.1908.03093, CoRR abs/1908.03093.

Park, H., Sjösund, L.L., Yoo, Y., Monet, N., Kwak, N., 2019b. SINet: Extreme lightweight portrait segmentation networks with spatial squeeze modules and information blocking decoder. http://dx.doi.org/10.48550/arXiv.1911.09099, CoRR abs/1911.09099.

Paszke, A., Chaurasia, A., Kim, S., Culurciello, E., 2016. ENet: A deep neural network architecture for real-time semantic segmentation. http://dx.doi.org/10.48550/arXiv.1606.02147, CoRR abs/1606.02147.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Vol. 32. Curran Associates, Inc., pp. 8024–8035. http://dx.doi.org/10.48550/arXiv.1912.01703.

Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J., 2017. Large kernel matters - Improve semantic segmentation by global convolutional network. http://dx.doi.org/10.48550/arXiv.1703.02719, CoRR CoRR abs/1703.02719.

Poudel, R.P.K., Liwicki, S., Cipolla, R., 2019. Fast-SCNN: Fast semantic segmentation network. http://dx.doi.org/10.48550/arXiv.1902.04502, CoRR abs/1902.04502.

Romera, E., Alvarez, J.M., Bergasa, L., Arroyo, R., 2017. ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation. IEEE Trans. Intell. Transp. Syst. PP, 1–10. http://dx.doi.org/10.1109/TITS.2017.2750080.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional networks for biomedical image segmentation. http://dx.doi.org/10.48550/arXiv.1505.04597, CoRR abs/1505.04597.

Santana, O.J., Hernández-Sosa, D., Smith, R.N., 2022. Oceanic mesoscale eddy detection and convolutional neural network complexity. Int. J. Appl. Earth Obs. Geoinf. 113, 102973. http://dx.doi.org/10.1016/j.jag.2022.102973.

Shelhamer, E., Long, J., Darrell, T., 2016. Fully convolutional networks for semantic segmentation. http://dx.doi.org/10.48550/arXiv.1605.06211, CoRR CoRR abs/1605.06211.

Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. J. Big Data 6 (1), 60. http://dx.doi.org/10.1186/s40537-019-0197-0.

Taha, A.A., Hanbury, A., 2015. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. BMC Med. Imaging 15 (1), 29. http://dx.doi.org/10.1186/s12880-015-0068-x.

Wambugu, N., Chen, Y., Xiao, Z., Tan, K., Wei, M., Liu, X., Li, J., 2021. Hyperspectral image classification on insufficient-sample and feature learning using deep neural networks: A review. Int. J. Appl. Earth Obs. Geoinf. 105, 102603. http://dx.doi.org/10.1016/j.jag.2021.102603.

Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G.W., 2017. Understanding convolution for semantic segmentation. http://dx.doi.org/10.48550/arXiv.1702.08502, CoRR CoRR abs/1702.08502.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., Xiao, B., 2019a. Deep high-resolution representation learning for visual recognition. http://dx.doi.org/10.48550/arXiv.1908.07919, CoRR CoRR abs/1908.07919.

Wang, Y., Zhou, Q., Liu, J., Xiong, J., Gao, G., Wu, X., Latecki, L.J., 2019b. LEDNet: A lightweight encoder-decoder network for real-time semantic segmentation. http://dx.doi.org/10.48550/arXiv.1905.02423, CoRR CoRR abs/1905.02423.

Wu, Y., He, K., 2018. Group normalization. http://dx.doi.org/10.48550/arXiv.1803.08494, CoRR abs/1803.08494.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J., 2018. Unified perceptual parsing for scene understanding. http://dx.doi.org/10.48550/arXiv.1807.10221, CoRR abs/1807.10221.

Xu, H., He, H., Zhang, Y., Ma, L., Li, J., 2023. A comparative study of loss functions for road segmentation in remotely sensed road datasets. Int. J. Appl. Earth Obs. Geoinf. 116, 103159. http://dx.doi.org/10.1016/j.jag.2022.103159.

Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J., Shen, F., 2022. Image Data Augmentation for Deep Learning: A Survey. arXiv, http://dx.doi.org/10.48550/ARXIV.2204.08610.

Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K., 2018. Denseaspp for semantic segmentation in street scenes. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3684–3692. http://dx.doi.org/10.1109/CVPR.2018.00388.

Yu, C., Gao, C., Wang, J., Yu, G., Shen, C., Sang, N., 2020. BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation. http://dx.doi.org/10.48550/arXiv.2004.02147, CoRR abs/2004.02147.

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N., 2018a. BiSeNet: Bilateral segmentation network for real-time semantic segmentation. http://dx.doi.org/10.48550/arXiv.1808.00897, CoRR abs/1808.00897.

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N., 2018b. Learning a discriminative feature network for semantic segmentation. http://dx.doi.org/10.48550/arXiv.1804.09337, CoRR abs/1804.09337.

Yuan, Y., Wang, J., 2018. OCNet: Object context network for scene parsing. http://dx.doi.org/10.48550/arXiv.1809.00916, CoRR abs/1809.00916.

Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J., 2017. ICNet for real-time semantic segmentation on high-resolution images. http://dx.doi.org/10.48550/arXiv.1704.08545, CoRR abs/1704.08545.

Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., 2016. Pyramid scene parsing network. http://dx.doi.org/10.48550/arXiv.1612.01105, CoRR abs/1612.01105.

Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N., Liang, J., 2018. UNet++: A nested U-Net architecture for medical image segmentation. http://dx.doi.org/10.48550/arXiv.1807.10165, CoRR abs/1807.10165.

Zhuang, J., 2018. LadderNet: Multi-path networks based on U-Net for medical image segmentation. http://dx.doi.org/10.48550/arXiv.1810.07810, CoRR abs/1810.07810, Withdrawn.

Chapter 5

Comparison of 3D CNNs for Volume Segmentation

Chapter 5 was published in "The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences", Volume XLVIII-1/W2-2023, 2023 as: **A COMPARATIVE STUDY OF DEEP ARCHITECTURES FOR VOXEL SEGMENTATION IN VOLUME IMAGES** [3]

Authors: Franz Wagner[1], Hans-Gerd Maas[1]

[1] Chair of Photogrammetry, TU Dresden, Dresden, Germany

Publication history: Submitted on March 31, 2023; accepted on May 23, 2023; revised on July 14, 2023; published on December 14, 2023.

Full reference: Wagner, F.; Maas, H.-G. (2023). A COMPARATIVE STUDY OF DEEP ARCHITECTURES FOR VOXEL SEGMENTATION IN VOLUME IMAGES in: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVIII-1/W2-2023, p. 1667-1676, DOI: 10.5194/isprs-archives-XLVIII-1-W2-2023-1667-2023

Contribution:
Following the "CRediT" guidelines by Brand et al. [1] (https://doi.org/10.1087/20150211), the author is responsible for: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing – original draft, and visualization. All authors contributed to: writing – review & editing. Hans-Gerd Maas is responsible for: resources, supervision, funding acquisition, and project administration.

A COMPARATIVE STUDY OF DEEP ARCHITECTURES FOR VOXEL SEGMENTATION IN VOLUME IMAGES

F. Wagner[1,2], H.-G. Maas[1]

[1]Institute of Photogrammetry and Remote Sensing, TU Dresden, Germany
[2]franz.wagner@tu-dresden.de

KEY WORDS: Deep Learning, 3D Segmentation, 3D CNN, Tomography, Magnetic Resonance Imaging, CNN Comparison

ABSTRACT:

This study investigates the performance of eight different deep learning architectures for voxel segmentation in volume images. The motivation is to segment carbon in carbon reinforced concrete (CRC) in micro-tomography (µ-CT) data. Although there are many 3D convolutional neural networks (CNNs) available, it is not yet clear which one works best for these specific tasks. In this study, the authors compare the following networks: DenseVoxNet, HighResNet, Med3D, Residual 3D U-Net, 3D SkipDenseSeg, 3D U-Net, V-Net, and LV-Net. To provide a more general recommendation for selecting a neural network, three medical datasets were added in addition to the three CRC datasets to facilitate the selection of an appropriate network based on the dataset characteristics. The experiments emphasize the importance of the initial random state, used for example to initialize the network weights. On average, the 3D U-Net is the best generalizing CNN, followed by the Residual 3D U-Net and the 3D SkipDenseSeg. While the 3D U-Net is a good architecture to start with, the experiments show that it does not perform best on all domains. To achieve optimal results, the authors propose recommendations for selecting a 3D neural network based on the dataset attributes.

1. INTRODUCTION

To minimize the use of materials in buildings or other structures, carbon reinforced concrete can be used (Beckmann et al., 2021). As less concrete is used herein, the position of the carbon elements is of great importance. Therefore, we used a micro-tomography instrument (µ-CT), which consists of an X-ray source and a camera that takes a large number of projections as the object rotates. From these images, a volumetric reconstruction of the object is created. In these volumes, the task is to segment the carbon components, which can be done using convolutional neural networks (CNNs).

In 2012, (Ciresan et al., 2012) proposed a deep neural network approach to the task of pixel-wise classification, also known as semantic segmentation. Although the approach worked, it was relatively slow, so (Ronneberger et al., 2015) introduced the famous U-Net, which outperformed the previous approach on both speed and performance. Since then, many other segmentation-related neural networks have been published (e.g.: SegNet (Badrinarayanan et al., 2015), DeepLab (Chen et al., 2018), GCN (Peng et al., 2017) or UPerNet (Xiao et al., 2018)) to solve 2D segmentation problems. These types of CNNs have also been successfully applied to 3D data such as in computed tomography (CT), magnetic resonance imaging (MRI), or electron microscopy (EM). However, the performance of 3D convolutions, such as in the 3D U-Net (Çiçek et al., 2016), has further improved the segmentation accuracy on such datasets. (Mester et al., 2022) successfully applied a 3D U-Net to segment carbon rovings (bundles of single carbon fibers aligned in a grid) in concrete. However, they have not determined whether this is the optimal network for this task due to the lack of comprehensive reviews and comparisons of 3D CNNs. In medicine, 3D datasets are very common and thus 3D CNNs are well-known. However, most medical studies dealing with 3D data still use 2D CNNs for their analysis. According to (Singh et al., 2020) and (Niyas et al., 2022), only 8-11 % of published medical papers use 3D CNNs, although they would be suitable for this purpose. To fill this gap and to determine the best network for our research, this study compares 8 different 3D CNNs using the AiSeg project (https://gitlab.com/frawa/aiseg). The investigated networks are:

- DenseVoxNet (Yu et al., 2017)
- HighResNet (Li et al., 2017)
- Med3D (Chen et al., 2019)
- Residual 3D U-Net (Lee et al., 2017)
- 3D SkipDenseSeg (Bui et al., 2019)
- 3D U-Net (Çiçek et al., 2016)
- V-Net (Milletari et al., 2016)
- LV-Net (Lei et al., 2020)

To the best of our knowledge, there are currently no other comprehensive 3D CNN comparisons. With the exception of the Med3D publication, all networks listed were tested on a single domain only. Therefore, in this study, all networks were compared on 6 different datasets: Three public and three new datasets representing electron microscopy, CT, and MRI data. All datasets have their own challenges that networks must overcome.

2. DATASETS

A typical volumetric dataset consists of a large number of 2D images stacked on top of each other. What is represented by a pixel in a 2D image corresponds to a voxel in three-dimensional space. The datasets were acquired by different acquisition devices such as magnetic resonance imaging, electron microscopy and computed tomography. An overview of the devices used and the size of the datasets can be found in table 1. All datasets in use come with a dataset specific challenge that the networks need to handle. The new datasets for carbon rovings, concrete pores, and polyethylene fibers were created using Dragonfly (Object Research Systems (ORS), 2021).

Method	Dataset	Classes	Voxels (million)
CT	Carbon Rovings	2	9326.6
CT	Concrete Pores	2	9932.1
CT	PE Fibers	2	1486.4
EM	Brain Mitochondria	2	129.8
MRI	BraTS	4	9222.6
MRI	Head and Neck Cancer	9	975.2

Table 1: Overview of the datasets including their data size represented in voxels

2.1 Carbon Rovings

At RWTH Aachen University, Germany, a laboratory mortar extruder is used to integrate soft impregnated carbon textiles, also called rovings, into structural components (figure 1). (Mester et al., 2022) were interested in the surface area of the rovings inside the concrete in order to use them in the context of a coupled multiscale method. This is a challenging task due to the fact that µ-CT basically determines the physical density of a voxel, and physical densities of carbon and some concrete constituents are rather similar. The dataset created for this purpose (Wagner, 2023a) is represented by 3 different CT scans. The first two scans were sliced into multiple training and validation sub-volumes of size 128 x 256 x 256 (Depth (d) x Height (h) x Width(w)) voxels, while the third scan is used for testing only. The unaugmented dataset consists of 129 training and 33 validation volumes plus the test volume. The training data was augmented using random rotations around the X, Y and Z axes, resulting in 1134 volumes with a voxel size of 9.4 µm. This dataset contains rather large structures that should be segmented by the models.

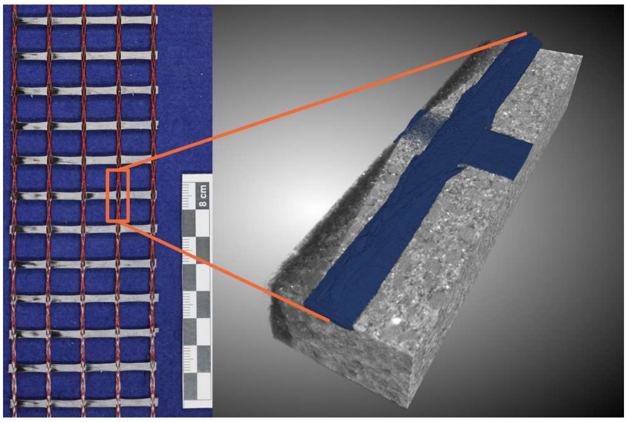


Figure 1: Carbon roving grid (left) and CT scan of a roving in concrete (blue, right) visualized with Dragonfly.

2.2 Concrete Pores

Segmenting pores in concrete is fairly straightforward task, for the obvious reason of different physical density. However, applying a simple thresholding may not be sufficient because the surrounding air is also segmented and some reconstructions may be extremely noisy depending on the power of the x-ray source used. Furthermore, depending on the reconstruction settings, the threshold has to be manually adjusted for each CT volume. Therefore, a new dataset for the segmentation of pores in concrete has been created (figure 2) (Wagner, 2023b). In its current state, it consists of 8 different CT scans, reduced to regions of interest. They were manually labeled and each scan was sliced into multiple sub-volumes of size 256 x 512 x 512 (d x h x w) voxels resulting in 148 training, 43 validation and 21 test volumes with different voxel sizes. This dataset contains very small to very large structures that the models should segment.

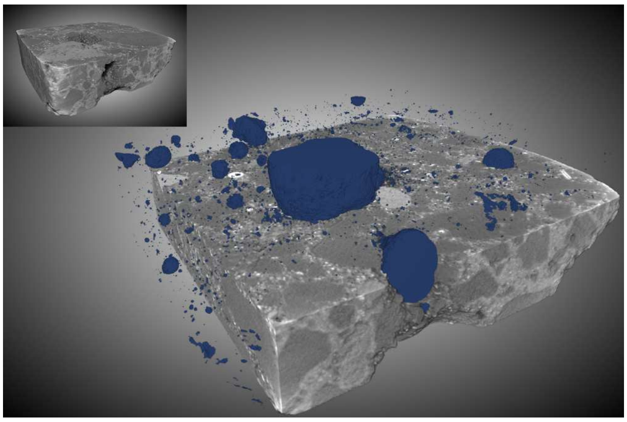


Figure 2: Miniature without labels and labeled pores (blue) visualized with Dragonfly.

2.3 Polyethylene Fibers

Segmentation of polyethylene fibers in strain-hardened cement-based composites is a very difficult task, since individual fibers of carbon or polyethylene bring the challenge that their density is very similar to one of quartz sand particles, resulting in almost identical gray values (Lorenzoni et al., 2020). The created PE fibers dataset (Wagner, 2023c) consists of only 3 spatially disjoint volumes of size 20 x 512 x 512 (d x h x w) voxels (figure 3) (voxel size: 4 µm). Since this is a rather small dataset, it was geometrically enlarged by combinations of rotation (using multiple angles), resizing, flipping, tilting, and squeezing using the AiSeg project. This is the only extensively augmented dataset, as training on so few images would lead to overfitting. A total of 397 training and 100 validation volumes were created. The original volumes were used as test volumes. The use of geometric augmentation results in different shapes for most of the new volumes. This dataset presents the challenge of using very little and only augmented data to predict real volumes. In addition, only very thin objects are included in this dataset.

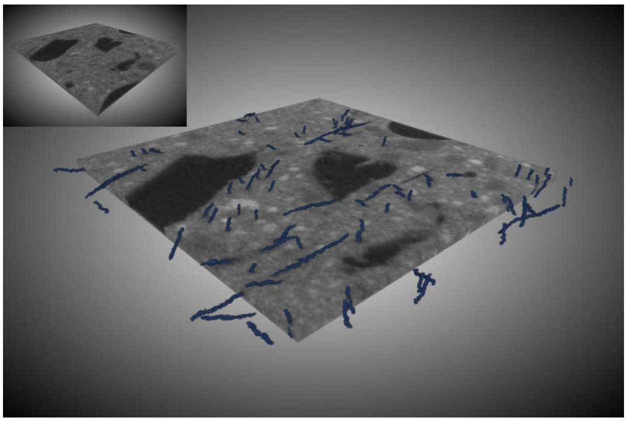


Figure 3: Miniature without labels and labeled volume containing fibers (blue) visualized with Dragonfly.

2.4 Brain Mitochondria

The Electron Microscopy dataset was created at EPFL in Lausanne to segment brain mitochondria in three-dimensional data

(Lucchi et al., 2013) (figure 4). It represents a small section of the hippocampal CA1 region of the brain and consists of two annotated volumes with a voxel size of 5 nm. The dimensions of the volumes are as follows: 165 x 768 x 1024 (d x h x w) voxels. The data represents rather large structures in a small dataset. However, the images are not perfectly aligned. The challenge for the models is to learn these features with limited data.

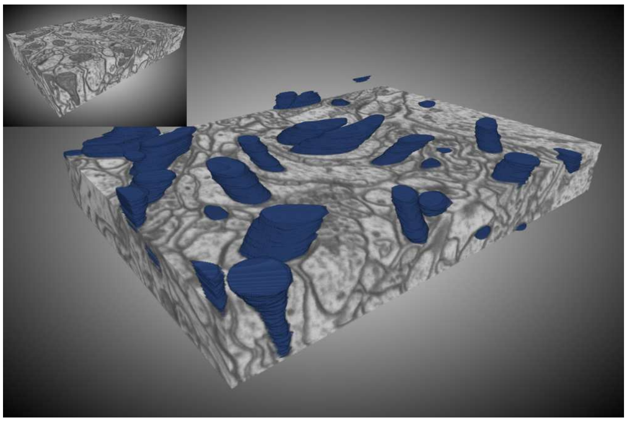


Figure 4: Miniature without labels and labeled brain mitochondria (blue) visualized with Dragonfly.

2.5 Brain Tumor Segmentation Challenge (BraTS)

The BraTS (2020) multimodal magnetic resonance imaging data were collected to develop and evaluate state-of-the-art methods for segmentation of brain tumors (namely gliomas). The published training data consists of 369 x 4 preoperative MRI scans of human brains, acquired at 19 different institutions. Each brain was imaged using native (T1), post-contrast T1-weighted (T1Gd), T2-weighted (T2), and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR) scans. The masks contain four labels: background, necrotic and non-enhancing tumor core (NCR/NET), peritumoral edema (ED) and GD-enhancing tumor (ET) (figure 5). For this study, all 1476 scans (369 x 4) were divided into 1033 training, 297 validation and 146 test volumes. Each volume is of size 155 x 240 x 240 (d x h x w) voxels with a voxel size of 1 mm. (Menze et al., 2015), (Bakas et al., 2017), (Bakas et al., 2018)
The challenges of this dataset are that it represents a multiclass problem and that the structures are intergrown.

2.6 Head and Neck Cancer

Radiation therapy is an important approach in the treatment of tumors. To prevent damage, the contours of tumors must be segmented with a high degree of confidence. The dataset of the Brain and neck cancer detection AAPM RT-MAC Grand Challenge 2019 (Cardenas et al., 2020), published in the Cancer Imaging Archive (Clark et al., 2013), aims to reduce common observer variability by making segmentation algorithms comparable. The MRI dataset consists of 55 scans, using T2-weighted images. 31 of them are used as training, 12 as validation and 12 as test data. Each volume has a size of 120 x 512 x 512 (d x h x w) voxels with a pitch of 0.5 mm per pixel and 2 mm per slice (2 x 0.5 x 0.5 mm³ (d x h x w)). The ground truth contains nine classes, resulting from a right and left subdivision and the background. The four main contours are: parotid glands, submandibular glands, level 2 and level 3 lymph nodes (figure 6). (Cardenas et al., 2019)

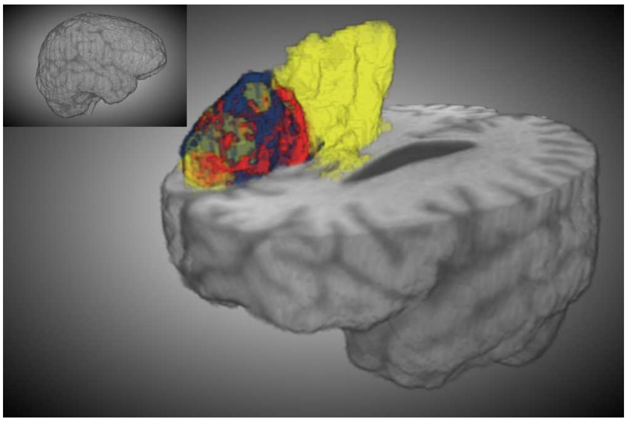


Figure 5: Miniature without labels and labeled BraTS data. Blue: Necrotic and non-enhancing tumor core; Yellow: Peritumoral edema; Red: Gadolinium-enhancing tumor visualized with Dragonfly.

The dataset consists of rather large structures. However, it has a different pitch in depth than in height and width. Also, some of the structures merge into each other.

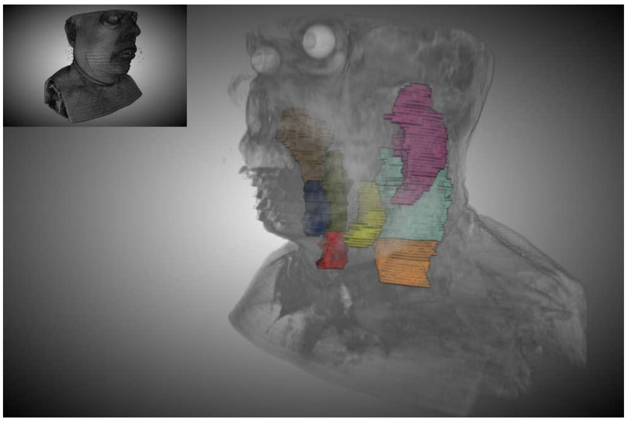


Figure 6: Miniature without labels and labeled organs at risk or tumors visualized with Dragonfly. Pink: submandibular gland (right); Teal: lymph node level 2 (right); Yellow: submandibular gland (right); Orange: lymph node level 3 (right)

3. METHODS

3.1 Data Preprocessing

When training 3D CNNs, most volumes do not fit into the video RAM (VRAM). Therefore, a training volume is divided into several equally overlapping sub-volumes. For example, in figure 7, the initial volume has the shape of 154 x 240 x 240 (depth (d) x height (h) x width (w)) voxels and is divided into 8 sub-volumes of 96 x 144 x 144 (d x h x w) voxels each. Each dataset presented is preprocessed according to this scheme. All datasets were, if they not already are, divided into training (needed to adjust the models weights), validation (needed to tune hyperparameters) and test (to evaluate the final model) data.

3.2 Hyperparameters

Hyperparameters are parameters that are set before a machine learning model is trained and affect how the model is trained and how it performs. Since we are comparing different architectures, the parameters for hidden layers, number of neurons

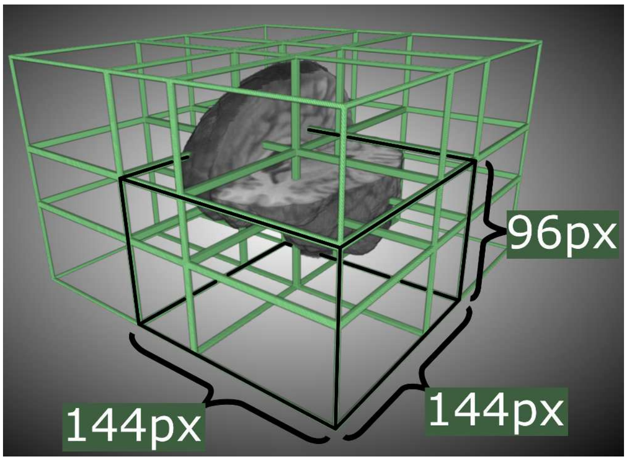


Figure 7: Example of a BraTS test volume with a size of 154 x 240 x 240 (d x h x w) voxels divided into 8 sub-volumes with a size of 96 x 144 x 144 (d x h x w) voxels. Black contours: one sub-volume.

in each layer and activation function are of course not identical. Due to the different dataset sizes, different input heights and widths (and depths for the fiber dataset) had to be used (table 2). Depending on the complexity and the number of parameters of the CNNs, the batch size is different for most networks. Since all batch sizes used are smaller than 16, group normalization was used as the normalization layer instead of batch normalization. The advantage of this technique is that it is more stable and produces better results than batch normalization on small batch sizes (Wu and He, 2018). The remaining hyperparameters related to training, such as total iterations, learning rate, optimizer, etc., are set the same to ensure a fair comparison. To evaluate the performance during training and validation, the cross-entropy loss (eq. 1) was used on a multi-class problem. For a binary segmentation, the binary cross entropy loss (eq. 2) was used. The optimizer was set to Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001.

Dataset	Height and Width	Depth	GPUs
Rovings	128	64	8
Pores	128	64	8
Fibers	256	16	4
Mitochondria	128	64	4
BraTS	128	64	8
Head Neck Cancer	128	64	4

Table 2: Overview of the datasets, their input dimensions and used GPUs during training.

$$L_{multi\ class} = -\sum_{c=1}^{C} y_{o,c} \log(p_{o,c}) \tag{1}$$

$$L_{binary} = -(y \log(p) + (1-y)\log(1-p)) \tag{2}$$

where: L = Loss value
p = Probability: observation o is of class c
y = ∈[0, 1]; indicator if classification c is correct
C = Number of classes

3.3 Metrics

During the training of a neural network, its performance was monitored using the loss (section 3.2) and the accuracy, which refers to the ratio of correctly classified voxels to all voxels in a volume. During training, the weights were adjusted iteratively in such a way that the loss value gets minimized. Using the validation loss and accuracy, the following statements can be derived:

- Low loss, low accuracy: many small errors
- Low loss, high accuracy: very little errors - best case
- High loss, low accuracy: many big errors - worst case
- High loss, high accuracy: very little but big errors

However, accuracy is not a reliable measure: Consider a volume of size 10 x 10 x 10 voxels containing only 50 voxels that belong to the foreground. If the network predicts every voxel to be background, the accuracy is still 95%, even though it failed badly. Therefore, we used the common *DICE* coefficient, also called the overlap index or F1-score (Taha and Hanbury, 2015), to measure the equality between the ground truth and the segmentation. Using the True Positives (TP), False Positives (FP) and False Negatives (FN), it is calculated by:

$$DICE = \frac{2TP}{2TP + FP + FN} \tag{3}$$

In the case of a multi-class problem (*BraTS* and *Head and Neck Cancer*), the *DICE* per class, calculated using eq. 3, and the class-biased mean (*mDICE*, eq. 4) will be be provided. For a general comparison, we also present a weighted mean *DICE* (*wDICE*, eq. 6). The weighting ensures that the score represents a global performance rather than a class-biased result. The weight is calculated by dividing the voxels (N) representing a class (c) by all voxels in the current volume (v).

$$mDICE = \frac{\sum_{c=1}^{n} DICE_c}{n} \tag{4}$$

$$w_c = \frac{\sum_v^t N_{c_v}}{\sum_v^t N_v} \tag{5}$$

$$wDICE = \sum_{c=1}^{n} w_c \cdot DICE_c \tag{6}$$

where: w_c = weighting per class
t = total volumes
n = all classes

During testing, the test volumes were subdivided as explained in section 3.1. Therefore, a voxel is represented by 2 or more logits at the overlapping regions. A logit is the direct output of the CNN at a single voxel. Furthermore, due to the convolutions and missing information at the edges, the output of a network is less certain at the corners than at the center of the prediction. Therefore, a common practice is to apply a 3D Gaussian weighting. In MONAI (Medical Open Network for AI, a PyTorch-based, open-source framework for deep learning) (Cardoso et al., 2022), a default sigma of $\sigma = input\_size \cdot$

0.125 is used, which depends on the input dimension. This results in very different weightings, e.g. from 0.00038 (input dimension of 128) to 0.61 (input dimension of 512) at the corners. Instead, we decided to weight the logits in such a way that the values at the corners are always weighted $1/3$ to those in the center (eq. 7). After weighting and combining all the subvolumes, the final prediction was evaluated against the ground truth.

$$\sigma = \sqrt{\frac{-0.5 \cdot (d^2 \cdot h^2 \cdot w^2)}{\log \frac{1}{3}}} \qquad (7)$$

3.4 Experimental Design

The experiments were performed on a High Performance Computing (HPC) system. Each training had 12 cores at 2.0 GHz and 60 GB RAM. Depending on the size of the dataset, 4 or 8 NVIDIA A100-SXM4 GPUs (40 GB VRAM) were used to train a network (table 2). To save computational time, reproducibility was disabled for all trainings since "deterministic operations are often slower than non-deterministic operations" (Paszke et al., 2019). This causes the values of the final weights to vary between two training runs due to different initial random states. The random state refers to the seed value used by random number generators within a machine learning algorithm. Therefore, the experiment is divided into two parts: In the first part, a rough estimation of the standard deviation (σ) of the *DICE* coefficient is performed using the head and neck cancer dataset (section 2.6). The performance of the networks is then further evaluated using the remaining five datasets.

Since the head and neck cancer dataset is quite small, the computation time is expected to be short, so this dataset was chosen to compute the standard deviation. However, since the training is still very computationally intensive, each network was trained only three times on the head and neck cancer dataset. For the other five datasets, the networks were trained only once.

With the exception of the PE fiber dataset (section 2.3), we did not perform strong augmentation, as this would change the properties of the datasets and thus make interpretation difficult. However, since this is a common practice, we refer to the AiSeg project for better and more robust results. The software is able to perform 3D offline and online augmentation as described in (Wagner et al., 2023).

4. RESULTS AND DISCUSSION

This section is divided into two parts. First, a rough estimate of the standard deviation regarding the *DICE* coefficient is made using the Head and Neck Cancer Dataset in dependence of the random initialization is conducted. Second, the performance of each network on the remaining five datasets is investigated. All neural networks were trained from scratch.

To make a rough estimate of the training duration, we approximated the duration as if we trained each dataset on a single A100-SXM4 GPU, resulting in over 101 days of training. If we had used a single RTX 3090, the training duration would have increased to almost a year (359 days).

4.1 Impact of Initial Random States (Head and Neck Cancer Dataset)

The purpose of this section is to make the reader aware that different initializations are likely to produce different results. To give a rough estimate, we trained all networks three times on the relatively small (table 1) Head and Neck Cancer dataset (section 2.6) to calculate an average and the standard deviation. Table 3 shows that the Med3D architecture with the ResNet10 backbone achieves the lowest σ with 0.08% and the V-Net the highest with 0.62%. Using this information, the results in the following sections should be treated with keeping these results in mind. Also, the standard deviation should decrease with the size of a dataset because there are many more volumes and therefore the variance of a dataset is likely to be higher.

Head and Neck Cancer **Network** (Backbone)	Parameters (million)	***wDICE* (%)** **Mean**	 $\boldsymbol{\sigma}$	 **Rank**
DenseVoxNet	1.7	96.54	0.32	14.
HighResNet	0.8	98.30	0.26	8.
Med3D (ResNet10)	17.3	98.61	0.08	1.
Med3D (ResNet18)	36.1	98.52	0.13	3.
Med3D (ResNet34)	66.5	98.34	0.34	7.
Med3D (ResNet50)	52.3	98.50	0.15	4.
Med3D (ResNet101)	91.3	98.12	0.27	9.
Med3D (ResNet152)	123.5	98.12	0.30	10.
Med3D (ResNet200)	132.7	98.10	0.10	11.
Residual 3D U-Net	141.2	98.43	0.34	6.
3D SkipDenseSeg	7.1	98.57	0.09	2.
3D U-Net	16.3	98.48	0.20	5.
V-Net	45.6	97.79	0.62	12.
LV-Net	12.2	97.56	0.60	13.

Table 3: Parameter count of all networks, mean testing *wDICE* coefficient of three training runs and standard deviation (σ) of the Head and Neck Cancer dataset.

In terms of the weighted performance, the Med 3D (ResNet 10) achieves the highest *wDICE*, closely followed by the SkipDenseSeg on this multiclass problem. Both networks have a small standard deviation of 0.08% and 0.09%, respectively, which covers the average difference of these two networks ($98.61\% - 98.57\% = 0.04\%$). Therefore, the assumption that the Med 3D (ResNet 10) performs best is not significant. For the Med3D backbones, it can be seen that fewer layers in the backbone are more powerful as fewer parameters need to be adjusted which is crucial for small datasets. The DenseVoxNet performs the worst of all the networks (96.54%).

Since this is a multiclass dataset, we also present the class-wise *DICE* in table 4 with the *mDICE* and their standard deviations in table 5 as the overall performance does not provide detailed information about the results. Compared to the other CNNs, the SkipDenseSeg was shown to perform best with an average of 1.39% (*mDICE*), although the achieved *DICE* scores are rather poor, which is reflected in figure 8. However, this is most likely due to the fact that the dataset is rather small, which is also supported by the comparison of the standard deviations per class. The more voxels a class contains, the lower the standard deviations become. In the following, the number of voxels associated with each class is shown, presenting the bias towards the background class which on its own consists of 98.71% of all voxels. Background: 962 594 506, C1: 590 569, C2: 589 474, C3: 1 121 325, C4: 1 140 637, C5: 2 207 852, C6: 2 379 147, C7: 2 226 010, C8: 2 326 160. The background class, has a low σ while all other classes have high standard deviations.

4.2 Carbon Rovings

The Carbon Rovings dataset is the second largest (table 1) and aims at segmenting quite large structures. The experiments have shown that 3D U-Net performs best on such elements (98.56%, table 6, figure 9). The gap to the second place (DenseVoxNet) is 0.39%, which is outside the standard deviation range of the two networks (DenseVoxNet: 0.32%; 3D U-Net:

Mean *DICE* (%) per class (Head and Neck Cancer Dataset)

Network (Backbone)	**B**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	***mDICE***
DenseVoxNet	97.37	33.80	28.70	20.83	14.37	33.43	26.77	45.77	47.43	38.72
HighResNet	98.93	45.37	37.70	35.70	34.27	49.93	44.83	59.53	59.60	51.76
Med3D (ResNet10)	**99.23**	48.30	43.20	**43.63**	**37.97**	53.53	**48.67**	59.57	58.07	54.69
Med3D (ResNet18)	**99.23**	44.07	38.13	36.43	35.83	48.63	44.27	47.27	47.87	49.08
Med3D (ResNet34)	99.13	34.57	31.47	32.20	27.90	42.70	37.00	43.13	41.40	43.28
Med3D (ResNet50)	99.20	36.40	39.33	36.77	29.30	46.80	45.30	48.57	56.03	48.63
Med3D (ResNet101)	99.00	26.23	28.53	26.57	19.43	31.60	30.00	38.10	45.03	38.28
Med3D (ResNet152)	99.07	20.13	27.37	27.40	23.83	34.87	28.93	24.03	19.77	33.93
Med3D (ResNet200)	98.97	18.13	13.53	26.47	23.47	34.00	31.60	41.23	31.93	35.48
Residual 3D U-Net	99.13	48.23	**52.33**	33.13	27.97	44.33	38.87	58.63	59.47	51.34
3D SkipDenseSeg	99.17	**49.20**	49.37	41.17	35.63	**55.67**	46.97	63.93	63.57	**56.08**
3D U-Net	99.07	46.97	46.83	35.00	33.63	47.53	48.50	**65.97**	**69.83**	54.81
V-Net	98.67	25.33	16.73	19.37	25.47	32.40	33.53	37.00	42.33	36.76
LV-Net	98.40	12.77	7.30	32.33	23.53	31.77	32.80	44.90	41.77	36.17

Table 4: Mean testing *DICE* coefficient per class on the Head and Neck Cancer Dataset. Labels: background (B), submandibular glands (left (C1) and right (C2)), level 2 (left (C3) and right (C4)) and level 3 (left (C5) and right (C6)) lymph nodes and parotid glands (left (C7) and right (C8)).

σ (%) per class (Head and Neck Cancer Dataset)

Network (Backbone)	**B**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
DenseVoxNet	0.32	10.43	8.89	1.62	**0.81**	3.05	2.68	**0.51**	8.21
HighResNet	0.21	**2.12**	**2.11**	6.38	3.71	3.68	3.97	5.78	3.54
Med3D (ResNet10)	**0.06**	6.50	5.10	**0.42**	2.35	1.01	**1.59**	2.85	6.16
Med3D (ResNet18)	**0.06**	4.79	6.43	4.05	2.71	**0.61**	7.96	4.29	17.17
Med3D (ResNet34)	0.31	9.95	11.92	4.73	4.41	8.25	8.49	5.95	11.69
Med3D (ResNet50)	0.10	9.77	12.55	4.91	2.27	2.19	5.12	13.50	4.68
Med3D (ResNet101)	0.30	6.95	11.14	8.86	1.72	1.57	1.81	18.47	8.10
Med3D (ResNet152)	0.32	9.98	16.46	3.64	5.71	5.23	15.90	13.32	24.72
Med3D (ResNet200)	0.12	16.06	4.53	3.82	3.27	4.77	4.64	3.24	13.16
Residual 3D U-Net	0.29	10.37	6.76	5.28	6.25	9.02	6.04	8.60	7.78
3D SkipDenseSeg	**0.06**	5.65	2.70	3.14	3.22	2.36	3.20	4.25	2.28
3D U-Net	0.21	10.46	11.11	8.15	7.14	7.12	3.83	1.76	**1.77**
V-Net	0.47	13.95	21.46	7.62	8.06	17.08	13.79	31.60	9.58
LV-Net	0.46	6.50	5.84	14.12	11.94	15.63	19.21	16.30	3.47

Table 5: Standard deviation of the *DICE* coefficient per class on the Head and Neck Cancer Dataset. Labels: background (B), submandibular glands (left (C1) and right (C2)), level 2 (left (C3) and right (C4)) and level 3 (left (C5) and right (C6)) lymph nodes and parotid glands (left (C7) and right (C8)).

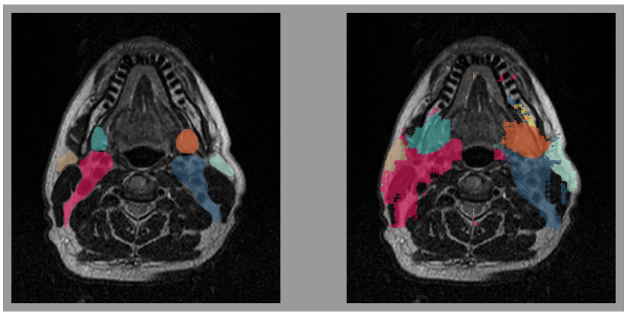


Figure 8: Visualization of ground truth (left) and segmentation (right) of the Head and Neck Cancer dataset using the 3D SkipDenseSeg. Orange: left submandibular gland; teal: right submandibular gland; yellow: left level 2 lymph node (not present in GT); gray: right level 2 lymph node (not present in GT and segmentation); blue: level 3 lymph node left; magenta: level 3 lymph node right; mint: left parotid gland; beige: right parotid gland.

0.20%). However, since this is a different dataset, the results of section 4.1 are not appropriate. Furthermore, the dataset is quite large, and therefore it is to be expected that the standard deviation will be lower. For this reason, we assume that the 3D U-Net performs best in binary segmentation with large structures and sufficient amount of training data. Although the 3D SkipDenseSeg performs best on the head and neck cancer dataset (*mDICE*) and achieves the lowest loss on this dataset, the gap in *DICE* with respect to the test data is 2.3% to the 3D U-Net, resulting in the second worst performance on this dataset.

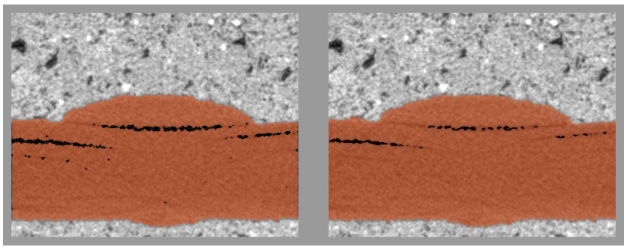


Figure 9: Visualization of ground truth (left) and segmentation (right) of the Carbon Rovings dataset using the 3D U-Net.

4.3 Concrete Pores

The Concrete Pores dataset consist of many tiny to large pores. Again, the 3D U-Net performs best, although not by a significant margin (gap to second: 0.11%, *DICE*: 87.66% (table 6, figure 10). On the second, the 3D SkipDenseSeg achieves a *DICE* of 87.55%, followed by the LV-Net, which achieves 86.95%. Although this dataset is similar in size to the Rovings dataset, all networks performed significantly worse. This is likely due to the fact that the pore structure is much more diverse in shape and size compared to a roving.

4.4 Polyethylene Fibers

For the Polyethylene Fibers dataset, the challenge is to predict tiny structures with limited data, which is a common problem for many researchers facing new domains. To overcome the problem of limited data, we used only geometrically augmented volumes to train and validate the networks. Testing was done

DICE (and *wDICE*) (%)

Network (Backbone)	**Head and Neck Cancer**	**Carbon Rovings**	**Concrete Pores**	**PE Fibers**	**Brain Mitochondria**	**BraTS**	**avg Rank**
DenseVoxNet	96.54	98.17	85.77	39.70	67.45	97.56	7.
HighResNet	98.30	96.09	84.85	46.38	67.61	98.24	6.
Med3D (ResNet10)	**98.61**	97.87	79.64	36.26	63.65	96.49	9.
Med3D (ResNet18)	98.52	97.36	80.41	37.83	63.47	96.55	13.
Med3D (ResNet34)	98.34	97.52	83.73	36.93	61.03	97.80	12.
Med3D (ResNet50)	98.50	97.37	82.93	40.29	70.97	97.88	4.
Med3D (ResNet101)	98.12	96.36	85.71	37.24	71.60	97.99	5.
Med3D (ResNet152)	98.12	97.39	75.61	41.15	65.94	97.58	10.
Med3D (ResNet200)	98.10	97.41	80.24	38.56	63.48	97.73	11.
Residual 3D U-Net	98.43	97.57	83.44	**63.49**	73.66	**98.59**	2.
3D SkipDenseSeg	98.57	96.28	87.55	52.98	69.78	97.98	3.
3D U-Net	98.48	**98.56**	**87.66**	58.45	76.38	98.08	1.
V-Net	97.79	97.05	72.10	16.21	63.74	96.47	14.
LV-Net	97.56	97.40	86.95	16.16	**78.89**	97.05	8.

Table 6: Testing *DICE* (and *wDICE* for multi-class datasets) coefficient regarding all networks and datasets. On the right hand side, the average rank is given, representing a mean ranking of the networks regarding all datasets.

Validation Loss

Network (Backbone)	**Head and Neck Cancer (avg)**	**Carbon Rovings**	**Concrete Pores**	**PE Fibers**	**Brain Mitochondria**	**BraTS**
DenseVoxNet	0.284	0.046	0.114	0.094	0.252	0.599
HighResNet	0.327	0.051	0.178	0.089	0.298	0.670
Med3D (ResNet10)	0.209	0.050	0.289	0.141	0.336	0.773
Med3D (ResNet18)	0.232	0.062	0.268	0.152	0.370	0.738
Med3D (ResNet34)	0.248	0.060	0.253	0.149	0.396	0.748
Med3D (ResNet50)	0.239	0.059	0.197	0.114	0.266	0.701
Med3D (ResNet101)	0.314	0.053	0.223	0.127	0.279	0.690
Med3D (ResNet152)	0.293	0.061	0.235	0.134	0.318	0.690
Med3D (ResNet200)	0.291	0.067	0.173	0.125	0.297	0.651
Residual 3D U-Net	0.315	0.048	0.139	0.083	0.211	0.576
3D SkipDenseSeg	0.228	**0.037**	0.158	0.082	0.281	0.571
3D U-Net	**0.202**	0.041	**0.093**	**0.067**	**0.207**	**0.560**
V-Net	0.638	0.052	0.465	0.298	0.730	0.804
LV-Net	0.736	0.102	0.307	0.305	0.525	0.757

Table 7: Validation Loss regarding all networks and datasets.

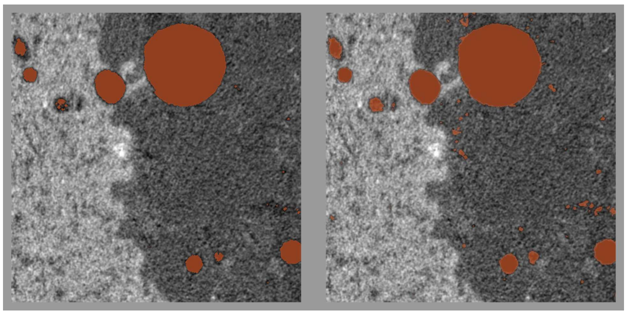


Figure 10: Visualization of ground truth (left) and segmentation (right) of the Concrete Pores dataset using the 3D U-Net.

on the unaugmented volumes, which explains the rather poor results of, for example, the V-Net and the LV-Net. We suspect a weakness in their architecture that makes them inefficient at learning generalized features on such data. The 3D-U-Net and its residual version are superior. In (Ronneberger et al., 2015), the U-Net was shown to perform well on small datasets, which has been proven in the third dimension as well. The Residual 3D U-Net significantly outperformed all other non U-Net architectures by a large margin (11.51%, table 6), and although it is the largest network in terms of parameters (141.2M), the inner architecture shows its efficiency. In total, only three networks were able to achieve a *DICE* of > 50% and can be considered useful for such problems: 1. Residual U-Net 3D, 2. 3D U-Net, 3.: 3D SkipDenseSeg. Even though the Residual 3D U-Net achieved a *DICE* of only 63.49% it reliably found all fibers in the test data (figure 11). The rather poor score results to the fact that fibers marked in the ground truth are very thin and maybe too conservative in size. The predicted structures are thicker and, on such a scale, this difference explains the achieved *DICE*.

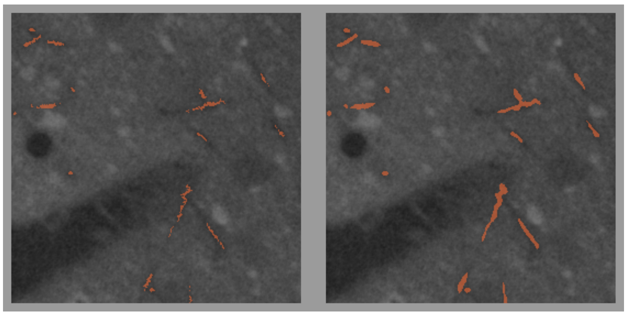


Figure 11: Visualization of ground truth (left) and segmentation (right) of the PE Fibers dataset using the Residual 3D U-Net.

4.5 Brain Mitochondria

The Brain Mitochondria dataset is quite small. Also, the training data is very similar to the test data, which is why the LV-Net, last place in the PE Fibers dataset test, performed best with a *DICE* of 78.89%. It seems that this architecture has a high potential to learn the features of the training dataset, but cannot adapt them to slightly different volumes. It requires a sufficient amount of training volumes to successfully learn and adapt features on unseen data. However, the loss is quite high with 0.525

(table 7). This happens when the network performs well in general, but produces a few large errors as shown in figure 12 (top right), where the network falsely predicts a large structure. This also supports the hypothesis that this architecture has a very narrow applicability. The second and third best models are the two U-Net variants, which again proves their usability on small datasets. However, the difference in *DICE* between the LV-Net and the 3D U-Net (second best) is still significant at 2.51% although the 3D U-Net was able to learn more features than the LV-Net when comparing the validation loss (table 7).

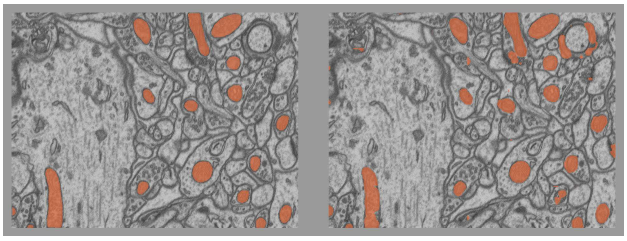


Figure 12: Visualization of ground truth (left) and segmentation (right) of the Brain Mitochondria dataset using the LV-Net.

4.6 BraTS

The large BraTS dataset contains complex structures to learn. However, all architectures perform quite well overall (*wDICE* from 96.47% to 98.59%, table 6), because the background class consists of the most voxels (98.7% background). Therefore, we also show the *DICE* per class as the overall performance is misleading, as already stated in section 4.1. The class distribution of voxels associated to each class is as follows: Background: 10 357 065 861, C1: 30 280 730, C2: 79 352 001, C3: 26 598 048. Table 8 indicates that the prediction of the single classes is a difficult task for all tested CNNs. The Residual 3D U-Net performs best (*mDICE*: 62.5%, visualization in figure 13), closely followed by the 3D SkipDenseSeg (*mDICE*: 59.5%). The worst network is the Med3D (ResNet10) with a *mDICE* of 42.6%.

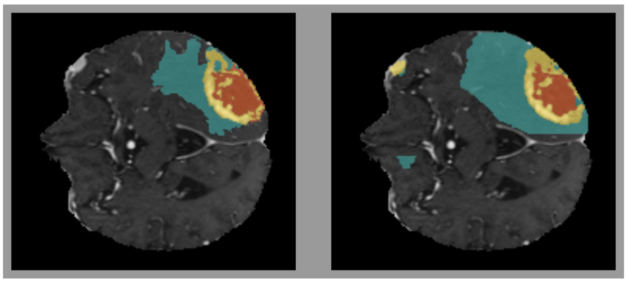


Figure 13: Visualization of ground truth (left) and segmentation (right) of the BraTS dataset using the Residual 3D U-Net. Orange: NCR/NET; teal: ED; yellow: ET

***DICE* (%) per class** (BraTS) **Network (Backbone)**	**B**	**NCR/NET**	**ED**	**ET**	***mDICE***
DenseVoxNet	98.2	45.4	33.2	43.2	55.0
HighResNet	98.8	48.2	38.9	**51.1**	59.2
Med3D (ResNet10)	97.3	24.6	21.4	27.0	42.6
Med3D (ResNet18)	97.3	36.9	21.0	26.8	45.5
Med3D (ResNet34)	98.6	35.0	27.7	30.9	48.0
Med3D (ResNet50)	98.6	46.4	30.4	33.7	52.3
Med3D (ResNet101)	98.7	36.2	32.0	37.1	51.0
Med3D (ResNet152)	98.3	39.1	26.1	38.1	50.4
Med3D (ResNet200)	98.5	38.6	28.4	28.2	48.4
Residual 3D U-Net	**99.2**	**60.2**	**43.4**	47.0	**62.5**
3D SkipDenseSeg	98.6	53.9	34.7	50.9	59.5
3D U-Net	98.8	39.8	33.0	45.3	54.2
V-Net	97.2	29.8	23.1	36.3	46.6
LV-Net	97.8	43.8	25.1	45.4	53.0

Table 8: Testing *DICE* coefficient per class on the BraTS dataset. Labels: background (B), necrotic and non-enhancing tumor core (NCR/NET), peritumoral edema (ED) and GD-enhancing tumor (ET).

5. CONCLUSION

The experiments have emphasized the importance of the initial random state. The minimum and maximum standard deviations ranged from 0.08% (Med3D (ResNet10)) to 0.62% (V-Net) on the Head and Neck Cancer dataset. In addition to choosing the correct hyperparameters, this can have a significant impact on the performance, and a final network should be trained several times to achieve the best results. In table 6, the rankings of all networks have been averaged across all datasets. Overall, the standard 3D U-Net performed best. The next best CNNs are the Residual 3D U-Net and the 3D SkipDenseSeg, although it never performed best on any domain. The comparisons have shown that the domain has an impact, albeit less than expected, and that the U-Net variants are fairly general architectures applicable to any dataset. However, although the 3D U-Net or its residual version can be a good starting point, in some cases, other networks are superior. From our results, we propose the following recommendations for selecting a 3D neural network based on dataset attributes:

- Larger datasets with large, coherent structures: 3D U-Net or DenseVoxNet
- Larger datasets with tiny to large structures: 3D U-Net or 3D SkipDenseSeg
- Larger datasets with complex structures: Residual 3D U-Net, HighResNet or 3D U-Net
- Medium datasets with large and very similar structures: 3D U-Net
- Small datasets with simpler, intergrown structures: a Med3D version or 3D SkipDenseSeg
- Very small datasets with very thin structures: Residual 3D U-Net or 3D U-Net

Although the LV-Net performs best on the Brain Mitochondria dataset, the experiments suggest that it has a weakness in its architecture that reduces its generalizability. Therefore, we cannot confidently recommend its use on such medium-sized datasets and have decided to recommend the 3D U-Net instead. The experiments also showed that neither the V-Net nor most of the Med3D versions (ResNet: 34, 50, 101, and 200) could achieve a top 3 result in any domain.

6. ACKNOWLEDGEMENTS

The research work presented in the publication has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB/TRR 280; Projekt-ID: 417002380
The authors are grateful to the Center for Information Services and High Performance Computing [Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)] at TU Dresden for providing its facilities for high throughput calculations.

REFERENCES

Badrinarayanan, V., Handa, A., Cipolla, R., 2015. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling. *CoRR*, abs/1505.07293. https://doi.org/10.48550/arXiv.1505.07293.

Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J. S., Freymann, J. B., Farahani, K., Davatzikos, C., 2017. Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Sci. Data*, 4, 170117. https://doi.org/10.1038/sdata.2017.117.

Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A., Shinohara, R. T., Berger, C., Ha, S. M., Rozycki, M., Prastawa, M., Alberts, E., Lipková, J., Freymann, J. B., Kirby, J. S., Bilello, M., Fathallah-Shaykh, H. M., Wiest, R., Kirschke, J., Wiestler, B., Colen, R. R., Kotrotsou, A., LaMontagne, P., Marcus, D. S., Milchenko, M., Nazeri, A., Weber, M., Mahajan, A., Baid, U., Kwon, D., Agarwal, M., Alam, M., Albiol, A., Albiol, A., Varghese, A., Tuan, T. A., Arbel, T., Avery, A., B., P., Banerjee, S., Batchelder, T., Batmanghelich, K. N., Battistella, E., Bendszus, M., Benson, E., Bernal, J., Biros, G., Cabezas, M., Chandra, S., Chang, Y., et al., 2018. Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge. *CoRR*, abs/1811.02629. https://doi.org/10.48550/arXiv.1811.02629.

Beckmann, B., Bielak, J., Bosbach, S., Scheerer, S., Schmidt, C., Hegger, J., Curbach, M., 2021. Collaborative research on carbon reinforced concrete structures in the CRC/TRR 280 project. *Civil Engineering Design*, 3(3), 99-109. https://doi.org/10.1002/cend.202100017.

Bui, T. D., Shin, J., Moon, T., 2019. Skip-connected 3D DenseNet for volumetric infant brain MRI segmentation. *Biomedical Signal Processing and Control*, 54, 101613. https://doi.org/10.1016/j.bspc.2019.101613.

Cardenas, C. E., Mohamed, A. S. R., Yang, J., Gooding, M., Veeraraghavan, H., Kalpathy-Cramer, J., Ng, S. P., Ding, Y., Wang, J., Lai, S. Y., Fuller, C. D., Sharp, G., 2020. Head and neck cancer patient images for determining auto-segmentation accuracy in T2-weighted magnetic resonance imaging through expert manual segmentations. *Medical Physics*, 47(5), 2317-2322. https://doi.org/10.1002/mp.13942.

Cardenas, C., Mohamed, A., Sharp, G., Gooding, M.and Veeraraghavan, H., Yang, J., 2019. Data from AAPM RT-MAC Grand Challenge 2019. The Cancer Imaging Archive. https://doi.org/10.7937/tcia.2019.bcfjqfqb.

Cardoso, M. J., Li, W., Brown, R., Ma, N., Kerfoot, E., Wang, Y., Murrey, B., Myronenko, A., Zhao, C., Yang, D., Nath, V., He, Y., Xu, Z., Hatamizadeh, A., Myronenko, A., Zhu, W., Liu, Y., Zheng, M., Tang, Y., Yang, I., Zephyr, M., Hashemian, B., Alle, S., Darestani, M. Z., Budd, C., Modat, M., Vercauteren, T., Wang, G., Li, Y., Hu, Y., Fu, Y., Gorman, B., Johnson, H., Genereaux, B., Erdal, B. S., Gupta, V., Diaz-Pinto, A., Dourson, A., Maier-Hein, L., Jaeger, P. F., Baumgartner, M., Kalpathy-Cramer, J., Flores, M., Kirby, J., Cooper, L. A. D., Roth, H. R., Xu, D., Bericat, D., Floca, R., Zhou, S. K., Shuaib, H., Farahani, K., Maier-Hein, K. H., Aylward, S., Dogra, P., Ourselin, S., Feng, A., 2022. MONAI: An open-source framework for deep learning in healthcare. https://doi.org/10.48550/arXiv.2211.02701.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L., 2018. 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. https://doi.org/10.1109/TPAMI.2017.2699184.

Chen, S., Ma, K., Zheng, Y., 2019. Med3D: Transfer Learning for 3D Medical Image Analysis. *CoRR*, abs/1904.00625. https://doi.org/10.48550/arXiv.1904.00625.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., Ronneberger, O., 2016. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. *CoRR*, abs/1606.06650. https://doi.org/10.48550/arXiv.1606.06650.

Ciresan, D., Giusti, A., Gambardella, L., Schmidhuber, J., 2012. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. *Advances in Neural Information Processing Systems*, 25.

Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., Moore, S., Phillips, S., Maffitt, D., Pringle, M., Tarbox, L., Prior, F., 2013. The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository. *Journal of Digital Imaging*, 26(6), 1045–1057. https://doi.org/10.1007/s10278-013-9622-7.

Kingma, D. P., Ba, J., 2015. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. https://doi.org/10.48550/arXiv.1412.6980.

Lee, K., Zung, J., Li, P., Jain, V., Seung, H. S., 2017. Superhuman Accuracy on the SNEMI3D Connectomics Challenge. *CoRR*, abs/1706.00120. https://doi.org/10.48550/arXiv.1706.00120.

Lei, T., Zhou, W., Zhang, Y., Wang, R., Meng, H., Nandi, A. K., 2020. Lightweight V-Net for Liver Segmentation. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1379-1383. https://doi.org/10.1109/ICASSP40776.2020.9053454.

Li, W., Wang, G., Fidon, L., Ourselin, S., Cardoso, M. J., Vercauteren, T., 2017. On the Compactness, Efficiency, and Representation of 3D Convolutional Networks: Brain Parcellation as a Pretext Task. *Information Processing in Medical Imaging*, 348–360. https://doi.org/10.1007/978-3-319-59050-9$_2$8.

Lorenzoni, R., Curosu, I., Paciornik, S., Mechtcherine, V., Oppermann, M., Silva, F., 2020. Semantic segmentation of the micro-structure of strain-hardening cement-based composites (SHCC) by applying deep learning on micro-computed tomography scans. *Cement and Concrete Composites*, 108, 103551. https://doi.org/10.1016/j.cemconcomp.2020.103551.

Lucchi, A., Li, Y., Fua, P., 2013. Learning for Structured Prediction Using Approximate Subgradient Descent with Working Sets. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 1987-1994. https://doi.org/10.1109/CVPR.2013.259.

Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., Lanczi, L., Gerstner, E., Weber, M.-A., Arbel, T., Avants, B. B., Ayache, N., Buendia, P., Collins, D. L., Cordier, N., Corso,

J. J., Criminisi, A., Das, T., Delingette, H., Demiralp, Ç., Durst, C. R., Dojat, M., Doyle, S., Festa, J., Forbes, F., Geremia, E., Glocker, B., Golland, P., Guo, X., Hamamci, A., Iftekharuddin, K. M., Jena, R., John, N. M., Konukoglu, E., Lashkari, D., Mariz, J. A., Meier, R., Pereira, S., Precup, D., Price, S. J., Raviv, T. R., Reza, S. M. S., Ryan, M., Sarikaya, D., Schwartz, L., Shin, H.-C., Shotton, J., Silva, C. A., Sousa, N., Subbanna, N. K., Szekely, G., Taylor, T. J., Thomas, O. M., Tustison, N. J., Unal, G., Vasseur, F., Wintermark, M., Ye, D. H., Zhao, L., Zhao, B., Zikic, D., Prastawa, M., Reyes, M., Van Leemput, K., 2015. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Trans. Med. Imaging*, 34(10), 1993–2024. https://doi.org/10.1109/TMI.2014.2377694.

Mester, L., Wagner, F., Liebold, F., Klarmann, S., Maas, H.-G., Klinkel, S., 2022. Image-based modelling and analysis of carbon-fibre reinforced concrete shell structures. S. Stokkeland (ed.), *Concrete innovation for sustainability: : held in Oslo, Norway, June 12-16, 2022*, fib proceedings, fib, 1631–1640.

Milletari, F., Navab, N., Ahmadi, S., 2016. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *CoRR*, abs/1606.04797. https://doi.org/10.48550/arXiv.1606.04797.

Niyas, S., Pawan, S., Anand Kumar, M., Rajan, J., 2022. Medical image segmentation with 3D convolutional neural networks: A survey. *Neurocomputing*, 493, 397-413. https://doi.org/10.1016/j.neucom.2022.04.065.

Object Research Systems (ORS), 2021. Dragonfly 2021.1. http://www.theobjects.com/dragonfly. Computer Software.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*, 8024–8035. https://doi.org/10.48550/arXiv.1912.01703.

Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J., 2017. Large Kernel Matters - Improve Semantic Segmentation by Global Convolutional Network. *CoRR*, abs/1703.02719. https://doi.org/10.48550/arXiv.1703.02719.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28.

Singh, S. P., Wang, L., Gupta, S., Goli, H., Padmanabhan, P., Gulyás, B., 2020. 3D Deep Learning on Medical Images: A Review. https://doi.org/10.48550/ARXIV.2004.00218.

Taha, A. A., Hanbury, A., 2015. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. *BMC Medical Imaging*, 15(1), 29. https://doi.org/10.1186/s12880-015-0068-x.

Wagner, F., 2023a. Carbon Rovings Segmentation Dataset. https://doi.org/10.34740/KAGGLE/DS/2920892.

Wagner, F., 2023b. Concrete Pores Segmentation Dataset. https://doi.org/10.34740/KAGGLE/DS/2921245.

Wagner, F., 2023c. Fiber Segmentation Dataset. https://doi.org/10.34740/KAGGLE/DS/2894881.

Wagner, F., Eltner, A., Maas, H.-G., 2023. River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs. *International Journal of Applied Earth Observation and Geoinformation*, 119, 103305. https://doi.org/10.1016/j.jag.2023.103305.

Wu, Y., He, K., 2018. Group Normalization. *CoRR*, abs/1803.08494. https://doi.org/10.48550/arXiv.1803.08494.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J., 2018. Unified Perceptual Parsing for Scene Understanding. *CoRR*, abs/1807.10221. https://doi.org/10.48550/arXiv.1807.10221.

Yu, L., Cheng, J., Dou, Q., Yang, X., Chen, H., Qin, J., Heng, P., 2017. Automatic 3D Cardiovascular MR Segmentation with Densely-Connected Volumetric ConvNets. *CoRR*, abs/1708.00573. https://doi.org/10.48550/arXiv.1708.00573.

Chapter 6

Segmentation of Carbon in CRC Using 3D Augmentation

Chapter 6 was published in "Buildings", Volume 13, Issue 9, 2023 (ISSN: 2075-5309) as: **Analysis of Thin Carbon Reinforced Concrete Structures through Microtomography and Machine Learning** [4]

Authors: Franz Wagner[1], Leonie Mester[2], Sven Klinkel[2], Hans-Gerd Maas[1]

[1] Chair of Photogrammetry, TU Dresden, Dresden, Germany
[2] Chair of Structural Analysis and Dynamics, RWTH Aachen University, Aachen, Germany

Publication history: Submitted on August 23, 2023; revised on September 15, 2023; accepted on September 18, 2023; published on September 21, 2023.

Full reference: Wagner, F.; Mester, L.; Klinkel, S.; Maas, H.-G. (2023) Analysis of Thin Carbon Reinforced Concrete Structures through Microtomography and Machine Learning in: *Buildings* 13 (9), 2399 – DOI: 10.3390/buildings13092399

Contribution:
Following the "CRediT" guidelines by Brand et al. [1] (https://doi.org/10.1087/20150211), both, the author and Leonie Mester contributed equally to: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing – original draft, and visualization. All authors contributed to: writing – review & editing. Sven Klinkel and Hans-Gerd Maas are responsible for: resources, supervision, funding acquisition, and project administration.

As this article was written in a collaborative manner with Leonie Mester, the sections to which the author contributed are listed:
Abstract: 50 % (Section 6.0), Introduction: 50 % (Section 6.1), Specimens: 100 % (Section 6.2.1), Microtomography: 100 % (Section 6.2.2), AI-Based Segmentation: 100 % (Section 6.2.3), Roving Extraction: 100 % (Section 6.2.4), Concrete Cover c_0: 25 % (Section 6.15, volume binarization), Microtomography: 100 % (Section 6.3.1), Deep Learning: 100 % (Section 6.3.2), Roving Extraction: 100 % (Section 6.3.3), Conclusions: 50 % (Section 6.4).
The software *Roving Surface Extractor* (https://gitlab.com/fra-wa/roving_surface_extractor) mentioned in the article was developed by Franz Wagner.

buildings

MDPI

Article

Analysis of Thin Carbon Reinforced Concrete Structures through Microtomography and Machine Learning

Franz Wagner [1,*,†], Leonie Mester [2,*,†], Sven Klinkel [2] and Hans-Gerd Maas [1]

[1] Institute of Photogrammetry and Remote Sensing, TUD Dresden University of Technology, 01062 Dresden, Germany
[2] Chair of Structural Analysis and Dynamics, RWTH Aachen University, 52074 Aachen, Germany
* Correspondence: franz.wagner@tu-dresden.de (F.W.); mester@lbb.rwth-aachen.de (L.M.)
† These authors contributed equally to this work.

Abstract: This study focuses on the development of novel evaluation methods for the analysis of thin carbon reinforced concrete (CRC) structures. CRC allows for the exploration of slender components and innovative construction techniques due to its high tensile strength. In this contribution, the authors have extended the analysis of CRC shells from existing research. The internal structure of CRC specimens was explored using microtomography. The rovings within the samples were segmented from the three-dimensional tomographic reconstructions using a 3D convolutional neural network with enhanced 3D data augmentation strategies and further analyzed using image-based techniques. The main contribution is the evaluation of the manufacturing precision and the simulation of the structural behavior by measuring the carbon grid positions inside the concrete. From the segmentations, surface point clouds were generated and then integrated into a multiscale framework using a parameterized representative volume element that captures the characteristic properties of the textile reinforcement. The procedure is presented using an example covering all necessary design steps from computed tomography to multiscale analysis. The framework is able to effectively evaluate novel construction methods and analyze the linear-elastic behavior of CRC shells.

Keywords: carbon reinforced concrete (CRC); computed tomography (CT); deep learning; 3D segmentation; 3D data augmentation; homogenization; shell structures

check for updates

Citation: Wagner, F.; Mester, L.; Klinkel, S.; Maas, H.-G. Analysis of Thin Carbon Reinforced Concrete Structures through Microtomography and Machine Learning. *Buildings* **2023**, *13*, 2399. https://doi.org/10.3390/buildings13092399

Academic Editors: Alexander Schumann and Frank Schladitz

Received: 23 August 2023
Revised: 15 September 2023
Accepted: 18 September 2023
Published: 21 September 2023

1. Introduction

Many studies have investigated carbon reinforced concrete (CRC) [1,2], and it is known that the tensile strength of carbon is significantly higher than that of steel [3,4], allowing for the construction of filigree components and further giving rise to the development of new construction methods. The research by Kalthoff et al. [5] explores such a method by using a laboratory mortar extruder (LabMorTex). They investigate impregnated glass and carbon reinforcements in various mortars and study the rheological properties of fresh concrete. Their proposed extrusion method can further deform extruded components into wave-like shapes, unlocking novel construction options [6,7]. However, such slender, shell-like structures require greater precision regarding the reinforcement compared to conventional construction methods. Furthermore, these methods necessitate the use of reliable evaluation methods with regard to their manufacturing accuracy.

To date, no guidelines exist for the design of textile-reinforced concrete components. However, numerous experimental tests have been performed to investigate the load-bearing behavior, and many different numerical methods have been used to capture the various effects that influence the load-bearing capacity. The present work focuses in particular on textile-reinforced shell structures. Within this investigation, the inner structure of such components was analyzed using a combination of state-of-the-art methods based on the findings in Mester et al. [8]. The authors proposed an approach to the analysis of CRC shells enhanced by image-based methods. In the current study, the methodology has been substantially improved in terms of automation and precision. The main contribution of this work is to study the position of the carbon grids and to simulate the load-bearing behavior of such textile-reinforced shell components. In addition, the accuracy of the extrusion process can be evaluated.

Since shell-like structures are characterized by large in-plane dimensions at small thicknesses (Figure 1), a multiscale method was used for the analysis. These methods investigate the structure at two scales: (1) the macroscopic scale, which describes the shell structure as well as its loading and boundary conditions, and (2) the mesoscopic scale, which represents the position and geometry of the textile reinforcement using a representative volume element (RVE). Figure 1 shows a textile-reinforced shell structure with an example inner structure to emphasize the large length differences (7 m span compared to only 6 cm thickness). Information on the manufacturing process and the actual reinforcement layout of this shell structure can be found in [9].

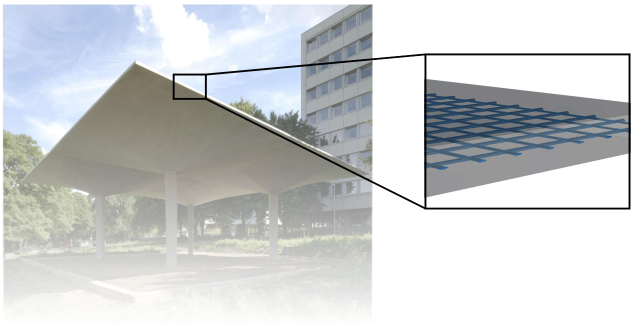


Figure 1. Pavilion made from carbon reinforced concrete with example inner structure. The dimensions of each shell are 7 m × 7 m × 6 cm.

To explore the inner composition of shell structures, four distinct CRC samples presented in [5–7] were studied, specifically focusing on the enclosed carbon grids. First, a micro-computed tomography (µ-CT) device was used to scan the samples and to obtain digital 3D representations of them. To analyze textile-reinforced shell structures using a multiscale model, the rovings have to be segmented from the volumetric reconstructions (Figure 2a). Since this task is not easily feasible without a huge interactive effort, a 3D convolutional neural network (CNN) was used. However, such a CNN needs a lot of training data, especially in 3D, to be applicable to new CT reconstructions and the contained rovings. Therefore, based on the 2D results and software architecture presented in [10], 3D data augmentation strategies for volume segmentation have been developed. Such methods allow for the improvement of the variance of a dataset without generating new training data. In the course of this study, the augmentation methods were intensively tested on a small dataset for semantic roving segmentation [11]. However, the data structure of a 3D segmentation, such as shown in Figure 2b, is not suitable to be directly integrated into a multiscale model. Therefore, the segmentation was further processed to derive a smoothed surface represented as a point cloud (Figure 2c).

The information obtained from the computed tomography images will be incorporated into the mesoscopic scale of the multiscale framework for analysis. Thus, the point cloud has to be transformed into a surface description suitable for analysis. As shown in [12,13], numerous methods have been presented for a variety of applications from different research fields. In contrast to polygonal meshes, the utilization of non-uniform rational B-splines (NURBS) for parametric surface descriptions offers the benefit of higher accuracy. Consequently, the geometric reconstruction of NURBS surfaces through imaging techniques is an important research area [14–16].

In the present contribution, scaled boundary isogeometric analysis (SBIGA) [17] is used for the analysis of the representative volume element. The method is boundary-oriented and utilizes NURBS surfaces, which are commonly used in computer-aided design, to describe the geometry. The use of isogeometric analysis [18,19] directly links design and analysis, which reduces approximation errors of the geometry in comparison to classical finite element methods.

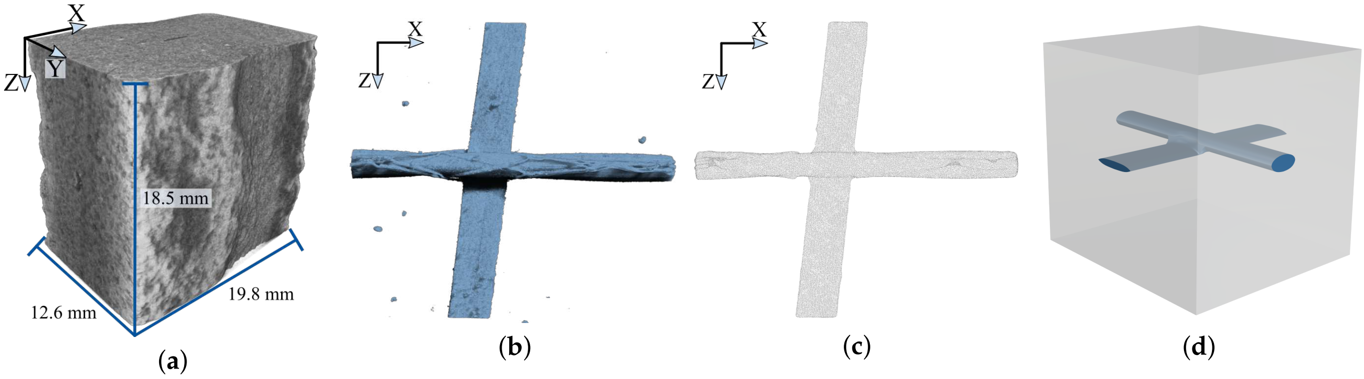


Figure 2. From computed tomography (CT) reconstruction to representative volume element. (**a**) A CT reconstruction containing a carbon roving. (**b**) A segmentation of the roving contained in (**a**). (**c**) Point cloud representing solely the roving from (**b**). (**d**) Representative volume element including the carbon roving. (**a**,**b**) were visualized using Dragonfly 2021.1 by Object Research Systems (ORS).

2. Materials and Methods

2.1. Specimens

Throughout this research, the components described in [5–7] were utilized and studied, focusing on the positioning of grid-aligned carbon rovings (carbon fiber bundles) in various mortar matrices. Figure 3a provides a schematic illustration of one such component. The uniformity of the carbon grid and the matrix allows for the definition of a single RVE that describes the main characteristic material properties. In this study, the RVE is embedded in a homogenization framework.

The components were extruded in warp direction, resulting in varying sizes of $6 \times X \times 1$ cm^3. For the investigation, a total of four samples, namely Samples A, B, C, and D (Figure 3), were extracted from four different components. All samples used in this study were uniformly cut to a size of $6 \times 2 \times 1$ cm^3.

The samples contain two different carbon grids consisting of multiple continuous rovings with slightly different yarn spacings. The first reinforcement is SITgrid044 VL by the company Wilhelm Kneitz Solutions in Textile GmbH, which will be referred to as Grid 1. The second grid is GRID Q43-CCE-21-E5 by the company solidian GmbH, which will be referred to as Grid 2. Besides the yarn spacing (Figure 4), the difference is that Grid 1 was coated using polystyrene, whereas epoxy resin was used for Grid 2. Grid 1 is contained in Sample A with a yarn spacing of 16.2 mm (warp) × 14.7 mm (weft) and in Sample B with a yarn spacing of 12.5 mm in both directions. Furthermore, the rovings of Grids 1 and 2 have different cross-sections, as shown in Table 1. In Sample B, a two-layer configuration of the textile was used. Sample C as well as Sample D contained Grid 2. A sanded surface was used for Grid 2 in Sample D to improve the composite behavior.

Table 1. Measured cross-sectional dimensions of the rovings contained in Grid 1 and Grid 2. At the intersection of the rovings in the warp and weft directions, Grid 1 has a height of ≈0.88 mm, and Grid 2 has height of ≈3.15 mm due to the coating.

Roving	Fiber Strand Grid 1		Fiber Strand Grid 2	
	Height in mm	Width in mm	Height in mm	Width in mm
Warp direction	0.55	1.41	1.41	2.13
Weft direction	0.29	2.35	1.02	2.51

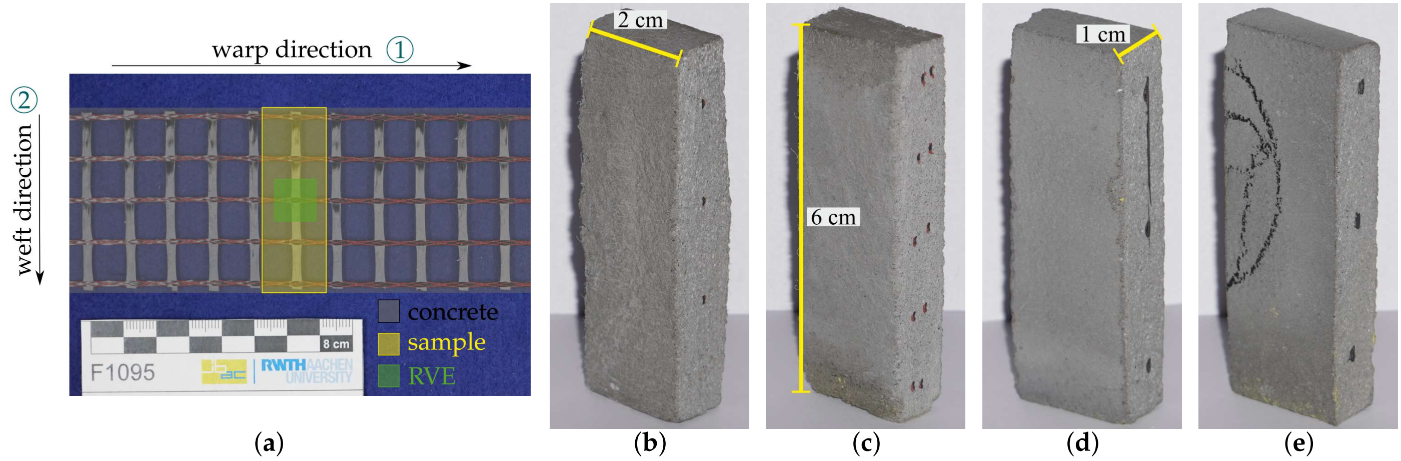


Figure 3. (**a**) Scheme of the cut-out region of a sample (yellow) with Grid 1 and the dimensions of a representative volume element (green). The component is indicated by the gray overlay. (**b**–**e**) From left to right: Samples A, B, C, and D with carbon reinforcement (dark spots). All samples are of roughly the same size.

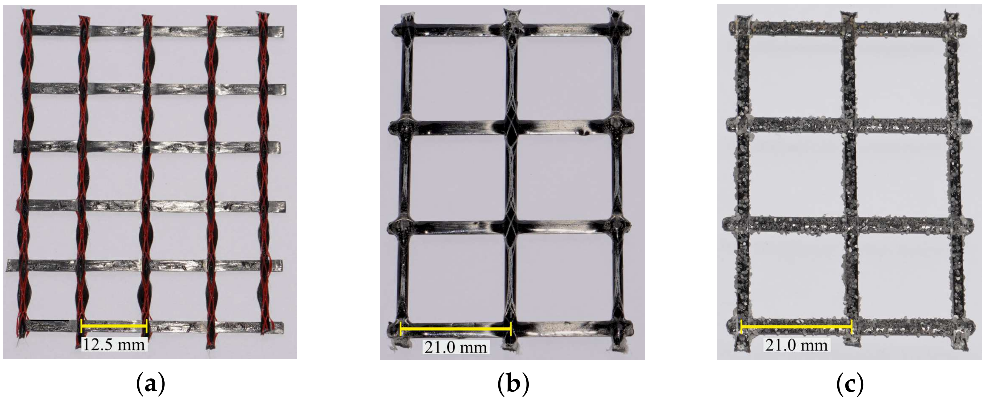


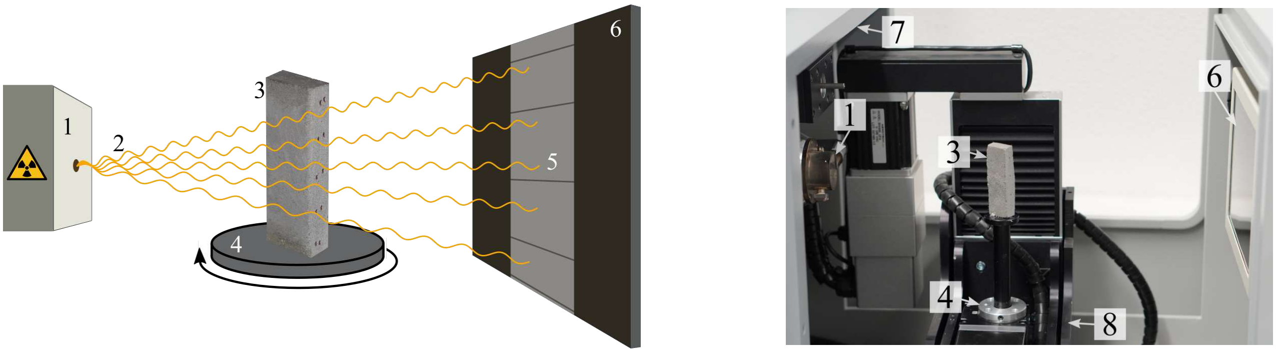
Figure 4. (**a**) Grid 1 with knitting thread (red), (**b**) Grid 2 with knitting thread (white), (**c**) Grid 2 with sanded surface.

2.2. Microtomography

To investigate the internal characteristics of objects, a µ-CT device can be used. Such a device (Figure 5) consists of an X-ray source (1), a rotating sample plate (4), and a detector (6). During the scanning process, a sample undergoes a complete 360° rotation, and at each rotation step, it is exposed to X-ray radiation, which is typically polychromatic, meaning that the radiation contains X-rays of multiple energies (2). As the X-ray beams traverse through the object being scanned (3), the detector captures the resulting projection image (5), which is generated by recording the X-ray attenuation data. In this study, the Procon CT-XPRESS, a µ-CT device, was utilized (Figure 5b). This instrument is designed to achieve a nominal resolution as low as 5 µm per voxel [20].

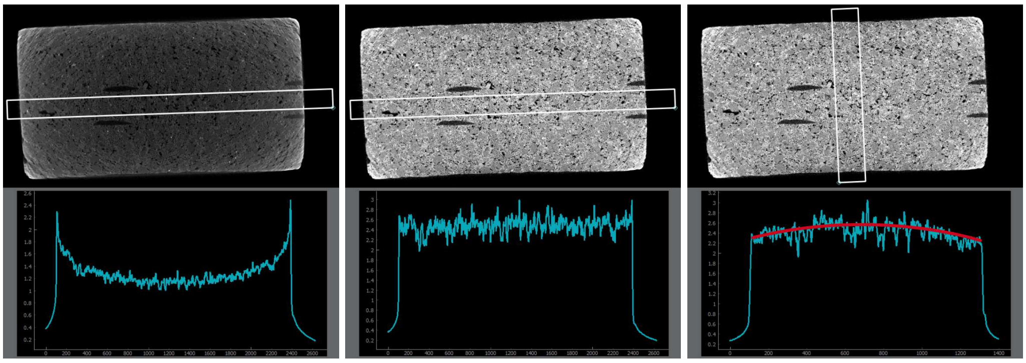
During a CT scan, the energy parameters, specifically the current and voltage, need to be adjusted based on the size of the specimen and the properties of the material being scanned. Materials with lower density and thickness require relatively little power, while denser materials necessitate higher energy levels. However, the use of higher power comes at the expense of introducing noise into the reconstructed images [21].

After concluding the CT scan, the gathered projection data are utilized for the reconstruction of a 3D volume using the software X-AID 2023.2 by MITOS GmbH. During the reconstruction, artifacts are likely to occur due to the physics of a CT. One error, which plays a role in this study, is the so-called beam hardening. Due to energy-dependent X-ray attenuation, the low energy X-rays (referred to as “soft X-rays”) of the generated spectrum are absorbed more quickly than the high energy X-rays (referred to as "hard X-rays") as they pass through a sample. As a result, the number of X-ray photons reaching an X-ray detector is not directly and linearly proportional to the thickness of the penetrated material. To correct this phenomenon, most reconstruction algorithms assume linear attenuation, which leads to a difference in coloration between the outer edge and inner material of the object. This discrepancy is known as the cupping effect [22]. In essence, the cupping effect causes the outer regions to appear differently shaded compared to the inner regions due to the beam hardening phenomenon as shown in Figure 6a. In Figure 6b, a linear correction has been applied. While the profile line appears relatively flat, there is a slight increase in brightness at the edges and especially at the corners. This effect is negligible for cylindrical samples but poses a problem for non-cylindrical samples, as present in this study. Figure 6c demonstrates an instance of the introduced error. The applied correction along the long side of the sample leads to darkening of the grayscale profile at the edges of the short side, resulting in a distinct color difference between the outer and inner regions of the sample.



(**a**) (**b**)

Figure 5. Scheme and computed tomography (CT) device (Procon CT-XPRESS) with 1: X-ray source, 2: X-rays, 3: sample, 4: rotating sample plate, 5: projection on 6: X-ray detector, 7: protective cover (lead), 8: movable sample table. (**a**) Scheme of a CT device; (**b**) Photo of the test setup.



(**a**) (**b**) (**c**)

Figure 6. Visualization of the beam hardening effect (cupping) and its correction during a computed tomography reconstruction process of Sample B using the reconstruction software X-AID 2023.2 (MITOS GmbH). Top: cross-sectional views; Bottom: grayscale profile. (**a**) Uncorrected reconstruction. The grayscale profile does not perfectly correlate with the specimen's thickness. (**b**) Correction of the cupping effect along the long side of the sample. (**c**) Correction of the cupping effect along the short side of the sample using the values of (**b**). Due to differences in thickness between the long and short sides, the correction becomes too strong, resulting in a non-linear profile (the red line should ideally be straight).

2.3. AI-Based Segmentation

In order to analyze the inner structure, visual examination may not be sufficient. In the present work, the carbon grids are extracted to generate a representative volume element of each grid and its position in the concrete. Therefore, the reinforcement has to be segmented first. Due to the fact that carbon has a different density than concrete and thus the grayscale representation is different from concrete, one possible approach seems to be a threshold approach to extract the rovings. In Figure 7a, a histogram is shown, clearly indicating the grayscale range of a carbon reinforcement in a horizontal slice of a reconstruction (Sample B). However, as explained in Section 2.2, the gray values of a CT reconstruction are likely to be inhomogeneous. Due to cupping effects, the grayscale range of the carbon reinforcement will vary from the outer to the inner region in the volume. In Figure 7b,c, the effect is shown. The binarized roving is more porous in the center than at the edges. Furthermore, small pores and slight noise cause other components in the reconstruction to have gray values similar to those of the carbon reinforcement. Since these issues make it difficult to reliably segment the carbon rovings, and since manual extraction for multiple rovings would be very time-consuming, machine learning techniques are used to automate the process.

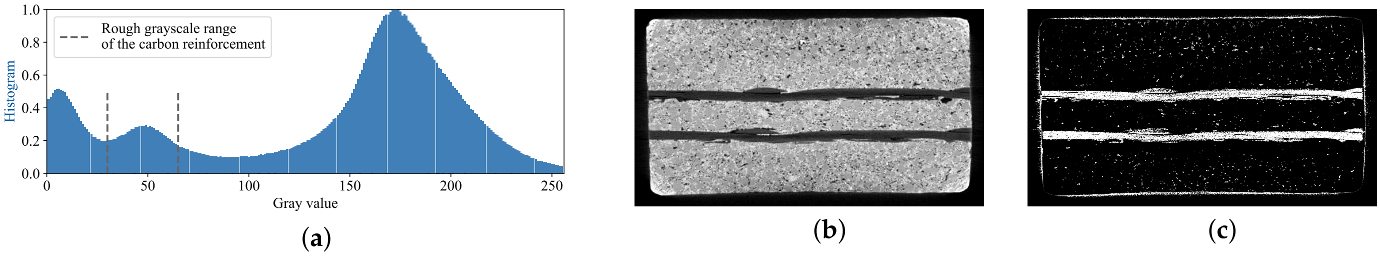


Figure 7. (**a**) Normalized histogram of a cross-sectional image (**b**) of Sample B. The grayscale range of the carbon reinforcement is highlighted. (**c**) The horizontal slice (**b**) was used to calculate the histogram. A threshold was applied based on the grayscale range in (**a**). The result is a binarized image: all values outside the range were set to 0 (black), while all voxels inside the range were set to 255 (white).

In the domain of image segmentation, neural networks have been successfully used since 2012 [23]. In 2015, the famous U-Net [24] marked a breakthrough due to its better performance while reducing the needed computational power. Following U-Net, many other CNNs were introduced [25–27], and 3D versions of segment volumes were proposed [28–30] as well as a 3D version of the U-Net [31]. In [32], eight different architectures for deep voxel classification were compared on multiple domains. It was shown that the 3D U-Net performed best on a carbon roving segmentation task, which is the reason it was chosen in this study. The 3D U-Net consists of an encoder–decoder structure, which is very common for image and volume segmentation networks. Since it is a very well-known and simple CNN, we refer to Çiçek et al. [31] for more detailed information on the architecture. However, prior to utilizing the neural network, it has to be trained, a process carried out through supervised learning that necessitated the availability of training data.

2.3.1. Training Data

During supervised training for volume segmentation, a CNN is guided to generate a segmentation of an input volume that matches the corresponding target ("ground truth"). However, such a dataset has to be created first. Fortunately, a Carbon Rovings Segmentation Dataset was introduced in [8] and subsequently improved and made publicly available in [32] on Kaggle (link can be found in reference [11]). This dataset comprises three CT scans containing three manually segmented samples using variants (different grid spacing and impregnation) of Grid 1 with a voxel size of 9.4 µm. Figure 8 showcases the two scans used for training and validation, and a third scan designated exclusively for testing.

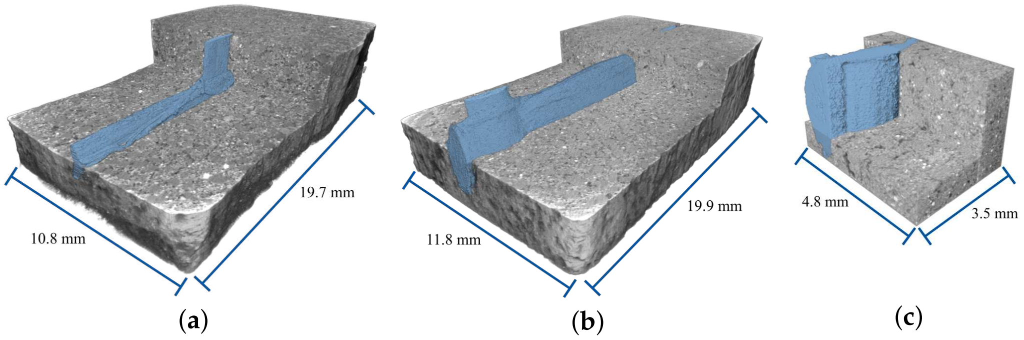


Figure 8. Visualization of the Carbon Rovings Segmentation Dataset [11] with manually segmented roving targets (light blue). The upper volumes have been reduced in depth to visualize the position of the rovings. (**a**) Training volume of Textile 1. (**b**) Training volume of Textile 2. (**c**) Training volume of Textile 3. Volumes (**a**,**b**) were split into multiple subvolumes and represent the training and validation data. The volume in (**c**) is used for testing. All textiles are variants of Grid 1 with different yarn spacings.

For training and validation purposes, the first two volumes were manually segmented and then divided into 162 subvolumes, each measuring 256 × 256 × 128 (height × width × depth) voxels. The division in training and validation data resulted in 129 training subvolumes and 33 validation subvolumes. However, a dataset of this size is rather small, and a well-generalized CNN that can be applied to different carbon grids is not an expected outcome unless data augmentation techniques are employed.

2.3.2. Augmentation

In addition to enabling the segmentation of Grid 2, data augmentation also helps to prevent overfitting. Overfitting occurs when the CNN performs exceptionally well on the training data but fails to generalize to unseen or new data. This happens when the model becomes too complex or has too many parameters relative to the size of the training data. As a result, the model memorizes the noise and specific patterns in the training data instead of learning the underlying general patterns, leading to a lower performance on new data [33].

Data augmentation is a technique to synthetically increase the diversity and quantity of the training dataset without collecting additional real-world data. This process helps the model to learn more robust and invariant features, improving its ability to handle various real-world scenarios and reducing the risk of overfitting to the limited training samples. Most deep learning libraries include basic image transformations such as flipping, rotating, scaling, and cropping [34,35], which are commonly utilized in various applications [36,37]. The Albumentations library [34] goes a step further and provides a more extensive set of augmentation techniques specifically designed for 2D images.

In this study, two methods were used, namely offline and online augmentation. The process of applying data manipulations before the training begins is referred to as offline augmentation. In this approach, an operator maintains control over the dataset, allowing for an inspection of the augmented images and masks in advance to ensure the appropriateness of the augmentations. However, a drawback of this method is the significant increase in required storage space. With numerous possible combinations of different augmentations for each image, storing all variations becomes impractical. In contrast, online augmentation eliminates the need for additional storage space since the operations are performed dynamically on the fly [33].

In a study by Wagner et al. [10], offline and online augmentation techniques for image segmentation were examined. The findings were presented alongside a software package called AiSeg (link in Data Availability Statement), designed to facilitate the training, testing, and utilization of 2D and 3D CNNs. This software also allows for the use of optimized offline and online augmentation pipelines for 2D images, which were extended to 3D in the course of this study. In total, 13 augmentations (flip, crop, resize, rotate, squeeze, tilt, blur, noise, brightness change, contrast manipulation, random erase, sharpen, and random darken) are supported and can be chained randomly. The software also allows many fine-tuning parameters to be set with regard to these augmentations, similar to the 2D versions, as explained in [10].

During model training, the high virtual RAM usage necessitates feeding only a subregion of a volume to the neural network in each iteration. As a result, the two augmentation approaches (offline and online) differ in their implementation. In offline augmentation, the original data are augmented, whereas in online augmentation, only a subvolume is augmented. The authors of [10] further emphasized that, for smaller datasets, offline augmentation proves beneficial, while larger datasets derive more advantages from online augmentation. The dataset used during this study is rather small, supporting the theory that offline augmentation may be beneficial. However, the creation of augmented 3D data needs a lot of storage space and therefore increases the training duration significantly compared to 2D images. Therefore, online augmentation was also investigated.

In addition to a training without any augmentation (for comparison reasons, Strategy 1), three training strategies with augmentation were tested. Strategy 2: To enhance performance on Grid 1, the training data were weakly offline augmented using only rotations. This strategy serves as a reference to [8], which used a similar approach. Strategy 3: To improve generalizability and to allow segmentation of Grid 2, strong offline augmentation was performed using suitable methods of the AiSeg software (link in Data Availability Statement). Strategy 4: To theoretically reduce training duration while increasing the dataset's variance, the 3D online augmentation was also carried out. In Figure 9, an example of an augmented subvolume is shown. The implementation details of the 2D augmentation pipelines can be found in [10], and the software design has been adapted to 3D.

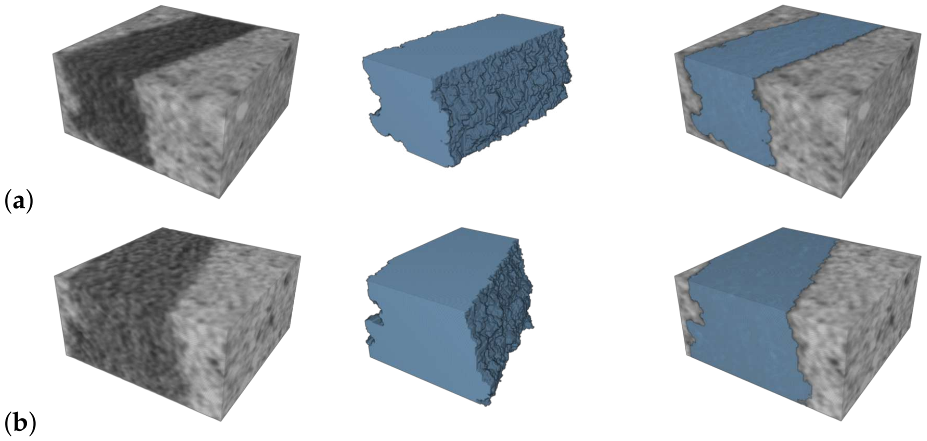


Figure 9. Example visualization of a training subvolume of size 128 × 128 × 64 voxels (height × width × depth) before (**a**) and after applied online augmentation (**b**). From left to right: subvolume, related ground truth, and combined versions.

2.3.3. Training and Metrics

The training of a CNN involves the process of iteratively adjusting the network's parameters to learn meaningful features from input data. At the beginning, the CNN's weights are randomly initialized. During each training iteration, a batch of input data is fed into the network, and its output is compared to the corresponding target using a loss function (e.g., cross-entropy). When iterating over the training data, the gradients of the loss with respect to the network's weights are then calculated through back-propagation. These gradients are used to update the weights using an optimization algorithm (e.g., stochastic gradient descent) to minimize the training loss. This process is repeated for multiple epochs until the performance of the network converges to a satisfactory level. After each epoch, the CNN in its current state is evaluated against the validation data, to estimate the validation loss on unseen data and to determine if overfitting has occurred. When both the training loss and the validation loss converge, training can be stopped, and the CNN can be used on unseen data.

Before initiating the training process, various parameters must be set, including the number of epochs, batch size, loss function, optimization algorithm, and others. These parameters, collectively known as hyperparameters, influence the model's training and performance outcomes. Below are the parameters that differ from the original implementation and are the same in the experiments conducted.

The batch size as well as the dimensions of the inputs are constrained by the available memory of the graphics card, which is a critical consideration when dealing with 3D data due to the increased CNN parameter count. Consequently, in this study, the batch size was capped at a maximum of three and an input size of 128 × 128 × 64 (height × width × depth). As a result, group normalization was favored over batch normalization as the normalization layer. This choice is driven by the fact that group normalization demonstrates greater stability and yields improved outcomes compared to batch normalization when handling small batch sizes (<16) [38].

Performance evaluation during both the training and validation phases was performed using the binary cross-entropy loss (Equation (1)). For optimization, the Adam optimizer was used due to its robustness over a range of hyperparameter configurations [39]. The initial learning rate was set to 0.001.

$$L = -w_c(y\log(p) + (1-y)\log(1-p)) \tag{1}$$

where: L = Loss value
w_c = Weight w of a class c
p = Predicted probability $p \in [0,1]$
y = Target class $y \in \{0,1\}$

To estimate the performance on unseen data, the *DICE* coefficient, also referred to as the overlap index or F1-score [40], was used. This coefficient quantifies the correspondence between the ground truth and the segmentation. Using the True Positives (*TP*), False Positives (*FP*) and False Negatives (*FN*), it is calculated by:

$$DICE = \frac{2TP}{2TP + FP + FN}. \tag{2}$$

During the testing and inference (application of a CNN) phases, the volumes were partitioned into overlapping subvolumes (50% overlap), since a complete volume does not fit in the GPU memory. As a result, voxels located in overlapping regions are represented by two or more logits. A logit is the direct output of the CNN for a single voxel. In addition, due to convolutions and the inherent loss of information at the edges, the network's output at the prediction corners is less reliable than in the central region. To mitigate this, it is common to apply a 3D Gaussian weighting [41]. In the CNN comparison (Section 2.3, [32]), a weighting of 1/3 of the voxels in the center to those in the corners was used. However, for improved smoothing across overlapping regions, the corner weights were set to 1/8 relative to the center weight. After applying these weights and aggregating all sub-volumes, the final prediction was generated. During a test phase, this volume was compared to the ground truth.

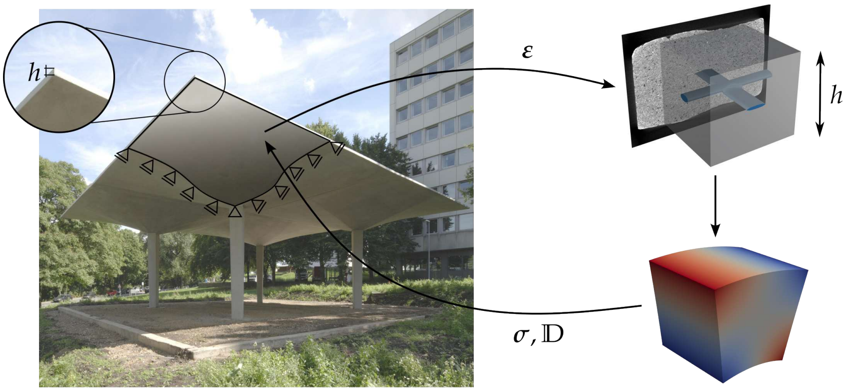
2.4. Roving Extraction

After successful training, the trained 3D U-Net can be used to process the 3D reconstructions obtained by tomography, as done in [8]. While the CNN is designed to accurately segment the roving, the presence of certain artifacts is likely. To address these artifacts and focus solely on extracting the roving and its surface, the authors used a connected components analysis method. This algorithm identifies clusters of adjacent voxels that are connected to each other, meaning they share a common corner or edge or that they are directly adjacent. In a single-layer carbon grid setup, the cluster consisting of the most connected voxels represents the roving. For this purpose, the authors of [8] also introduced software called the Roving Surface Extractor (link in Data Availability Statement).

Due to the fact that the knitting thread introduces unwanted complexity to the roving surface, a 3D Gaussian filter with a standard deviation of 10 was applied on the binary roving to smooth the surface and to remove possible artifacts introduced by the knitting thread. Afterwards, a point cloud describing only the surface was extracted from the smoothed 3D volume using the marching cubes algorithm [42] to significantly reduce the computational complexity of the next steps. This was used in the context of a multiscale model for the analysis of textile-reinforced shell structures.

2.5. Multiscale Modeling

Textile-reinforced shell structures are characterized by large in-plane dimensions and relatively small thicknesses, giving rise to the use of multiscale methods for analysis. These consider the structure on two scales, each of which employs different numerical methods; see Figure 10. The macroscopic scale describes the geometry, loading, and boundary conditions of the shell; see Figure 10a. The remaining characteristic properties of the structure, such as the geometry and position of the textile reinforcement, are described by a representative volume element. In the case of shell homogenization, the thickness of the RVE corresponds to the shell thickness h, as in Figure 10, which is a characteristic feature of the approach. Thus, the scale is referred to as mesoscale in the following (Figure 10b).



(a) Macroscale **(b)** Mesoscale

Figure 10. Schematic homogenization procedure for shell structures using computed tomography data on the mesoscopic scale. (Adapted with permission from Ref. [43]. 2023, L. Mester.)

Multiscale methods are often referred to as computational homogenization because they aim to model the macroscopic structure as homogeneous. However, they do not intend to develop full constitutive laws for the composite material. For carbon reinforced concrete, this has been done, for example, in [44,45].

Generally, computational homogenization offers a variety of methods and possible applications, and extensive overviews can be found, for example, in [46,47]. Homogenization with application to shell structures is discussed in [48–51].

The present contribution is based on the work of Gruttmann and Wagner [51] and employed a Reissner–Mindlin kinematic for the description of the macroscopic scale, which allows transverse shear effects to be taken into account. The scaled boundary isogeometric analysis [17] was used for the analysis of the RVE. The information obtained from the extracted roving, such as the orientation and position of the textile, was incorporated into the RVE. The two components, concrete and textile, were modeled separately. This allows separate material models to be used for each component. The effective material response for each point on the macrostructure is to be determined.

The procedure can be briefly outlined in the following steps: (1) For each macroscopic integration point, the strains $\boldsymbol{\varepsilon}$ were applied to the RVE using appropriate boundary conditions. (2) The mesoscopic boundary value problem was solved, and (3) the obtained stresses $\boldsymbol{\sigma}$ and effective material properties $\mathbb{D}$ were returned to the macroscopic scale. The procedure is shown schematically in Figure 10. This procedure is repeated for every macroscopic point. Here, the macroscopic shell structure is analyzed using finite elements; thus, the effective material response must be determined for each macroscopic integration point using the described procedure. Since the numerical problems on both scales are treated concurrently, the method is often referred to as FE^2 [52].

The mesoscopic and macroscopic scales must be coupled in an energetically consistent manner. Here, the Hill–Mandel condition [53] was used, which states that the virtual internal work on the mesoscopic scale averaged over the mid-surface of the RVE is equal to the macroscopic virtual work. From this, different boundary conditions for the lateral surfaces of the RVE can be derived; see, for example, [54]. Here, periodic boundary conditions were employed due to the repetitive structure of the reinforcement. However, the coupling of a two-dimensional structural element on the macroscopic scale and a three-dimensional continuum formulation on the mesoscopic scale is not trivial. The employed boundary conditions are presented in more detail in [55]. They require the constructed RVE to be symmetric with respect to the axes in the plane, as well as being point symmetric. This is an important requirement for the construction of a representative volume element suitable for analysis using computed tomography data.

2.6. Scaled Boundary Isogeometric Analysis

The mesoscopic representative volume element was analyzed using scaled boundary isogeometric analysis. Analogous to computer-aided design, volumes are described by means of their surfaces. Here, these were defined by NURBS, which can be used directly in the context of isogeometric analysis [18,19]. To transform the bivariate tensor product into a volumetric description, a scaling center C was introduced, using concepts known from the scaled boundary finite-element method [56]. The linear scaling direction, running from the scaling center to the boundaries, thus introduced the third direction needed for a volumetric description of a body. Thus, any volumetric body can be described by its boundary surface and a scaling center. The interior of the solid was then parameterized by a scaling parameter ξ, which runs from the scaling center (where $\xi = 0$) to the boundary (where $\xi = 1$). Correspondingly, the surface was discretized with the two parameters $0 \leq \eta, \zeta \leq 1$; see Figure 11. Therefore, any point in the domain can be described by:

$$\mathbf{X}(\xi, \eta, \zeta) = \hat{\mathbf{X}} + \xi\big(\tilde{\mathbf{X}}(\eta, \zeta) - \hat{\mathbf{X}}\big), \tag{3}$$

where $\tilde{\mathbf{X}}$ describes any point on the boundary, and $\hat{\mathbf{X}}$ denotes the position vector of the scaling center C.

Classically, the scaling center needs to be positioned within the kernel of the geometry to be visible from any point on the surface. This requires the geometry to be star-shaped. Recently, efforts have been made to overcome this limitation, for example, by using curved scaling lines, such as circular arcs or parabolic lines [57,58]. Alternatively, for the two-dimensional case, it has been shown that the scaling center can be positioned outside of the kernel and even outside of the geometry [59–61]. However, to date, these approaches are limited to two dimensions. Therefore, complex geometries are divided into several star-shaped domains in order to apply SBIGA. Some algorithms for the subdivision have been proposed [62], but again they are limited to two-dimensional domains. Consequently, an RVE representation is employed when the subdomains are known to be star-shaped.

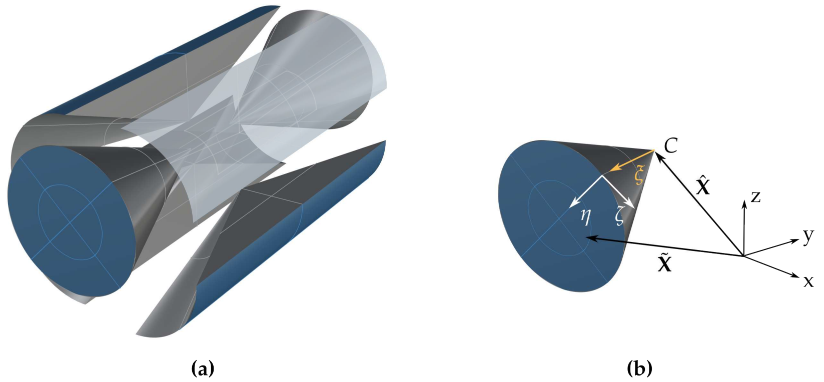


Figure 11. (**a**) Exploded view of a cylinder and (**b**) example discretization of a section.

2.7. Parameterized RVE and Definition of Characteristic Geometric Properties

From the first-order shell homogenization approach using scaled boundary isogeometric analysis on the mesoscale, two significant restrictions on the RVE have been derived. First, the use of periodic boundary conditions requires symmetry and point symmetry of the opposing RVE boundaries. In addition, the scaled boundary isogeometric analysis requires star-shaped geometries. Therefore, a parameterized RVE geometry considering a roving intersection has been developed and is shown in Figure 12. The main characteristics of the RVE are the grid opening in warp and weft directions l_1 and l_2, the concrete cover c_0, the RVE height h_{RVE}, as well as the roving dimensions that have been identified. From observations, the rovings were assumed to be elliptical, so their radii can be given by h_1/b_1 and h_2/b_2 for the weft and warp directions, respectively. Regarding their relative position, it is assumed that the lower edge of the ellipse describing the weft direction coincides with the center line of the elliptical cylinder describing the warp direction. The geometry is modeled using the Grasshopper® plug-in for the computer-aided design software Rhinoceros®. The parameterized model can be found on Zenodo [63].

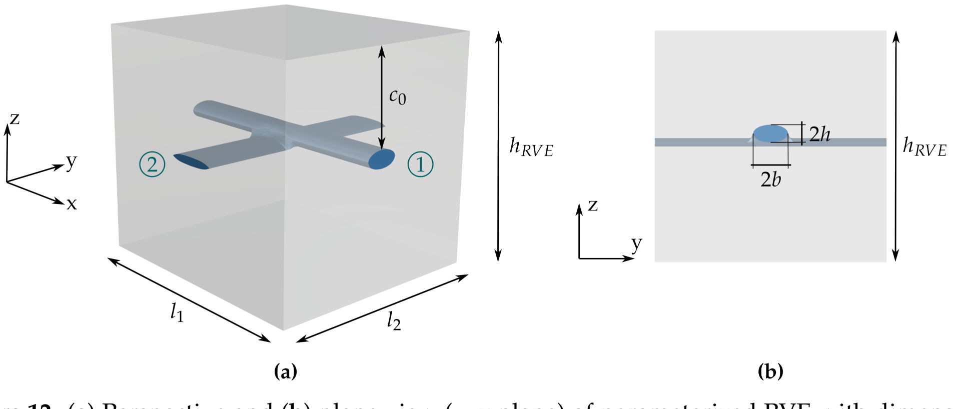


Figure 12. (**a**) Perspective and (**b**) plane view (y–z plane) of parameterized RVE with dimensions (1: warp direction, 2: weft direction).

These characteristics were obtained from the extracted roving geometry and data provided by the manufacturer. In the following, it is explained how to obtain the characteristic values.

Note that both constituents, concrete and textile, are assumed to be homogeneous. Therefore, any air inclusions or grains in the concrete are neglected.

2.7.1. Roving Dimensions b_1/b_2 and h_1/h_2

The roving geometry was approximated as elliptical with the radii h_1/b_1 and h_2/b_2 in the warp and weft directions, respectively. The width of the approximated roving is described as $2b$, and $2h$ corresponds to the height of the roving, as shown in Figure 12b.

To determine the radii, an averaged cross-sectional area was determined for each roving using the extracted 3D surface point cloud, as described in Section 2.4. A number of n_{slice} slices were evaluated along each roving axis, where each slice corresponds to a 2D image. Figure 13a shows an example of a cross-sectional image from computed tomography. From the extracted 3D point cloud describing the surface of the rovings, a 2D point cloud describing the outline of the roving was obtained for each cross-section. The concave hull algorithm, based on [64], was used to derive a polygon describing the outline of the roving. Figure 13b,c show the boundary point cloud from the roving extraction and its concave hull for the 2D image in Figure 13a.

From this, the centroid of each cross-section and the two radii were determined. The averaged properties of the roving are then obtained as:

$$\hat{A}_{roving} = \frac{\sum_i^{n_{slice}} A_i}{n_{slice}}, \quad \hat{\mathbf{x}}_{centroid} = \frac{\sum_i^{n_{slice}} \mathbf{x}_{centroid,i}}{n_{slice}}, \quad \hat{b} = \frac{\sum_i^{n_{slice}} b_i}{n_{slice}}, \quad \hat{h} = \frac{\sum_i^{n_{slice}} h_i}{n_{slice}}. \tag{4}$$

In general, it was not necessary to consider all slices obtained from CT data. The number of slices considered n_{slices} should be large enough to ensure a correct mean value. The area where two rovings intersect was not considered, since in these areas the point cloud describes the outline of both rovings.

From the averaged radii, the aspect ratio was obtained as $\kappa_{roving} = \hat{h}/\hat{b}$. Using the averaged cross-sectional area and the aspect ratio, the radii of the approximating ellipse can be determined from:

$$A = \hat{A}_{roving}, \quad \kappa = \kappa_{roving}, \quad b = \sqrt{\frac{A}{\pi \cdot \kappa}}, \quad h = \kappa \cdot b. \tag{5}$$

From these dimensions, the ellipse describing the cross-section of the roving was determined, with its centroid corresponding to the averaged centroid. Figure 14a shows the derived ellipse from only two concave hulls in plane view, while Figure 14b shows the perspective view of the derived cylindrical approximation and two used concave hulls.

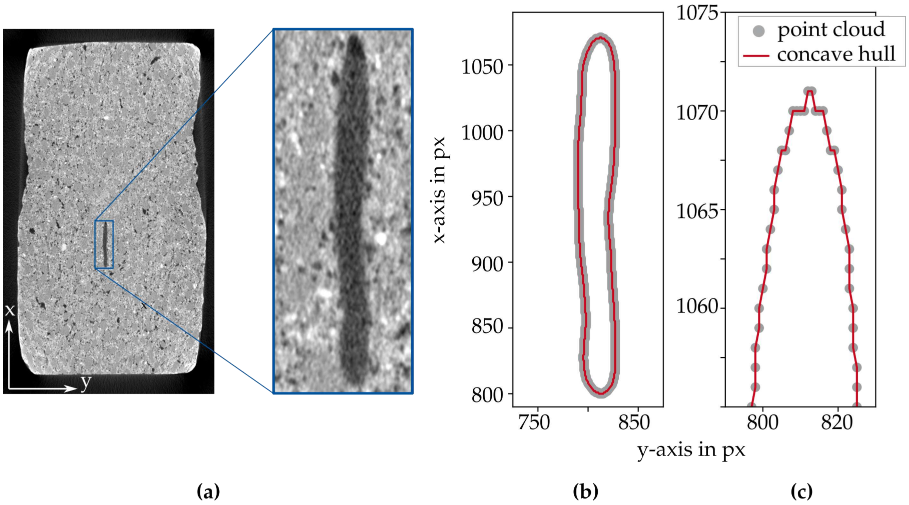


Figure 13. (**a**) Example cross-sectional image of Sample A with magnification of roving. (**b**) Corresponding point cloud and concave hull. (**c**) Magnification of point cloud and concave hull.

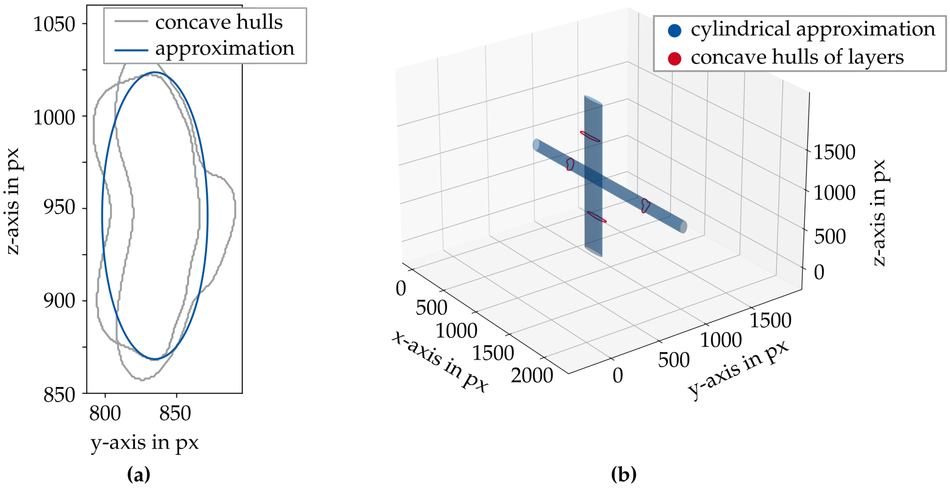


Figure 14. (**a**) Plane view of approximated ellipse from two concave hulls describing the roving. (**b**) Perspective view of derived elliptical cylinders with concave hulls used for approximation.

2.7.2. In-Plane Dimensions l_1/l_2

For the application of periodic boundary conditions in the framework of the homogenization approach, periodically repeating RVEs are assumed. Since only one intersection of the rovings was considered, the in-plane dimensions of the RVE can be directly correlated with the grid properties of the textile. Therefore, the dimensions l_1 and l_2 were taken from the data sheet of the manufacturer and correspond to the thread spacing (the distance between the center lines) in the warp and weft directions, respectively. For visualization, the in-plane RVE dimensions are depicted in Figure 3a.

2.7.3. Shell Thickness h_{RVE}

The RVE height h_{RVE} corresponds to the thickness of the concrete shell. It was determined prior to production. The average shell thickness of the scanned concrete sample can also be determined using computed tomography data. This can be beneficial for the evaluation of novel production methods, such as extrusion [5].

For this purpose, the cross-sectional images were binarized using a range-based threshold (e.g., as in Section 2.3, Figure 7) and the flood fill algorithm, i.e., all pixels within the boundary of the sample are set to 255 (white), and all pixels outside are set to 0 (black). The height h_{RVE} was determined by measuring the distance between the averaged upper ($y_{t,sample}$) and lower ($y_{b,sample}$) boundaries of the sample, indicated by the two red lines in Figure 15a. Note that 10% of the sample was neglected on each side. This choice was driven by the fact that the sample is not perfectly rectangular due to production imperfections. This procedure can be repeated for multiple images along the z-direction to determine an average thickness.

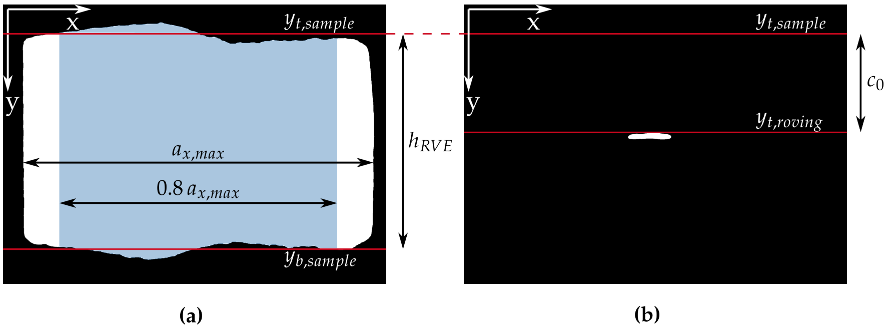


Figure 15. (**a**) Binary image of Sample A (pixels belonging to sample have value 255 (white)). (**b**) Binary image of the roving contained in Sample A (pixels belonging to extracted roving have value 255 (white)).

2.7.4. Concrete Cover c_0

Similar to the shell thickness, the concrete cover of the textile depends on the manufacturing process. Verification using CT data is important not only for the evaluation of new production processes, but also because the positioning of the textile within the shell is of interest for the load-bearing behavior, especially for thin shells. Due to the production process, the position of the textile can vary, for example, due to lifting or settling of the textile.

Again, the position can be determined using binary images. The procedure is similar to the one used for determining the shell thickness h_{RVE} (Section 2.7.3). The pixels belonging to the roving have the value 255 (white), and others have the value 0 (black), as shown in Figure 15b. The concrete cover was estimated using the distance from the averaged upper boundary (upper red line in Figure 15a) to the first white pixel in the y-direction (lower red line in Figure 15b). Again, this procedure was repeated for multiple 2D images along the sample to determine an average concrete cover.

2.7.5. Concluding Assumptions

To conclude, the assumptions made for the characterization of a representative volume element are briefly summarized:

1. Rovings are orthogonal to each other.
2. Only one intersection of rovings is considered.
3. Rovings are approximated as elliptical cylinders.

Although the present work examines extruded textile-reinforced specimens, the assumptions presented are also valid for other carbon reinforced concrete structures using grid-aligned carbon textile reinforcement, regardless of the construction technique. The main feature that justifies the assumptions is the periodically repeating textile reinforcement. Generally, any textile grid can be investigated; however, the chosen multiscale technique requires the RVE to be symmetric as well as point symmetric (see Section 2.5). Therefore, in the first step, the approach is limited to orthogonal reinforcement geometries.

3. Results and Discussion

3.1. Microtomography

The Procon CT-XPRESS (Section 2.2) was used to scan the four samples. The scanned region varies depending on the reconstruction settings and the placement of the samples relative to the X-ray source and detector. Figure 16 shows the the volumetric reconstructions of the four samples. Furthermore, Figure 17 displays cross-sectional images of Sample A, revealing a slight shift in the carbon grid introduced during the extrusion process. Figure 18 includes cross-sectional slices of all samples, indicating that more coating was used for Grid 2 compared to Grid 1. The coating is generally barely distinguishable from the roving. In Sample D, the coating is used to fix the sand to the surface.

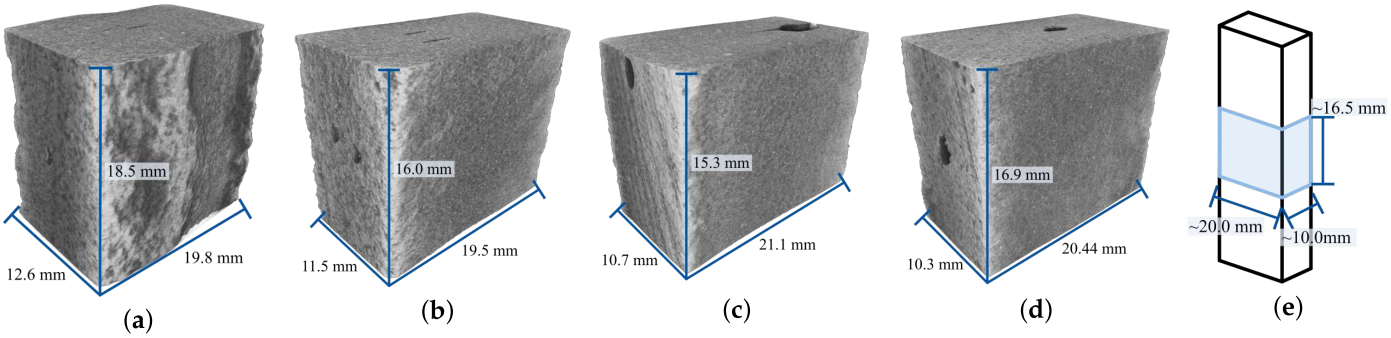


Figure 16. (**a**) Sample A (voxel size: 9.5 µm). (**b**) Sample B (voxel size: 8.9 µm). (**c**) Sample C (voxel size: 9.4 µm). (**d**) Sample D (voxel size: 9.4 µm). The dark spots in the reconstructions are the carbon grids. (**e**) Schematic of the scanned region.

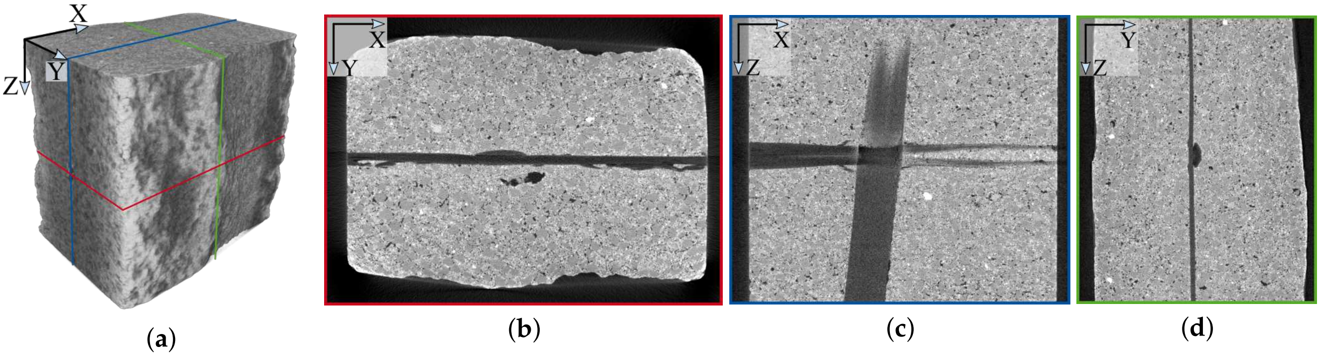


Figure 17. Cross-sectional images of a computed tomography reconstruction of Sample A. (**a**) Sample A with marked image positions. (**b**) X–Y-view with roving. (**c**) X–Z-view with roving. (**d**) Y–Z-view with roving.

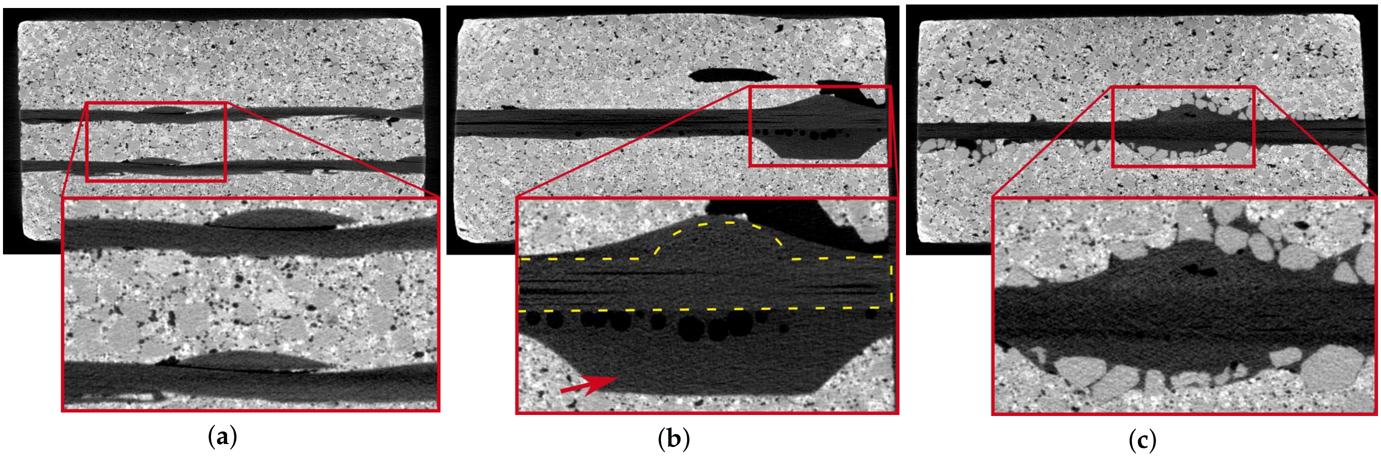


Figure 18. Cross-sectional images of Samples B, C, and D along with magnifications. (**a**) Sample B contains a two-layer configuration of Grid 1. (**b**) Sample C contains Grid 2 with a lot of coating. The roving is highlighted within the yellow contours. The coating is indicated by the red arrow. (**c**) Sample D contains Grid 2 with a sanded surface, clearly visible in the computed tomography scan.

3.2. Deep Learning

To extract the rovings, it is necessary to segment the four reconstructions using a trained 3D U-Net. As explained in Section 2.3.2, the network was trained using four strategies: (1) training with the standard dataset, (2) training with weak offline augmentation, (3) training with strong offline augmentation, and (4) training with online augmentation. The following methods were allowed during the augmentation:

- Weak offline augmentation:
  - Rotation around X, Y, and Z;
- Strong offline augmentation:
  - Rotation (around X, Y, and Z), resizing, flipping, tilting, squeezing, noise addition, blurring, sharpening, contrast manipulation, brightness manipulation;
- Online augmentation:
  - Offline: rotation around X and Y due to the difference in input height (128), width (128) and depth (64);
  - Online: rotation (around Z), resizing, flipping, tilting, squeezing, noise addition, blurring, sharpening, contrast manipulation, brightness manipulation.

Training 3D convolutional neural networks (CNNs) with various augmentation methods demands considerable computational resources due to the three-dimensional nature of the datasets. Therefore, this section starts by describing the hardware configuration. Subsequently, the results of the training experiments are provided.

3.2.1. Hardware and Training Duration

The 3D U-Net was trained on a high-performance computing (HPC) system. Throughout the training phases of Strategies 1, 2, and 3, 12 cores, 60 GB RAM, and 8 NVIDIA A100-SXM4 GPUs (40 GB VRAM each) were allocated. Since the online augmentation involved parallel batch processing, 24 cores (the maximum per node) were used. The training runs for the four strategies according to Section 2.3.2 were initially instructed to train the CNNs for 100 epochs. However, the online augmentation increased the variance of the datasets to such an extent that 200 epochs were required for the CNN to converge.

Table 2 shows the training time for all strategies. These training times are not representative for any other machine, but they do provide a rough comparison between these approaches. In [10], the online augmentation was as fast as a training without augmentation. However, in this study, 8 GPUs were used instead of 1. Therefore, with only 24 cores in total, each GPU had only three parallel processes to load and augment the input batches, which is clearly not enough for 3D.

Table 2. Computational time required to process 100 epochs, depending on the augmentation strategy. The dataset used for online augmentation has been increased as explained in Section 2.3.2.

Strategy	Training Volumes	Duration (100 Epochs)
1: No augmentation	516	01 h:09 m
2: Weak offline augmentation	3658	08 h:06 m
3: Strong offline augmentation	21,776	38 h:50 m
4: Online augmentation	1262	45 h:43 m

3.2.2. Training Results

In total, four CNNs were trained, each using a different strategy. The final results are shown in Table 3. As mentioned in Section 2.3.1, the dataset consists exclusively of variations of Grid 1. Therefore, high DICE coefficients and validation accuracies are expected. The accuracy represents the ratio of correctly classified voxels to all voxels in the validation dataset. The DICE values in Table 3 show that the generalization ability using Strategy 1 was the worst compared to the other strategies, even though the test data represent the same grid type.

Training with strong offline augmented data achieved the highest DICE, the highest validation accuracy, and the lowest validation loss. Strategy 4 has the highest loss but still a high accuracy, which means that the CNN generates many small errors.

Table 3. Training results of testing DICE and validation metrics.

Strategy	DICE in %	Validation Loss	Validation Accuracy
1: No augmentation	72.46	0.0493	98.88
2: Weak offline augmentation	98.62	0.0354	99.21
3: Strong offline augmentation	98.67	0.0239	99.34
4: Online augmentation	97.44	0.0853	98.92

Due to the high similarity between the training and validation data, no direct overfitting could be observed for all strategies during the training phases, as shown in Figure 19. The similarity is further emphasized by the convergence behavior of Strategies 1 to 3, since the validation loss was almost identical to the training loss. Therefore, splitting the dataset into training, validation, and test data seems to be ineffective.

A closer inspection of the graphs suggests that the CNNs using Strategies 1 and 2 may exhibit a tendency to overfit specifically on Grid 1 variations due to the strong correlation with the loss graphs. This may also be true for Strategy 3 but cannot be said with certainty. In contrast, only Strategy 4 introduced enough variability into the training data to create a notable distinction from the validation data. Consequently, the validation loss diverged from the training loss in this case, and both losses were higher when compared to Strategies 1 to 3.

However, the results lack information on the generalization ability of the networks with respect to different carbon grids.

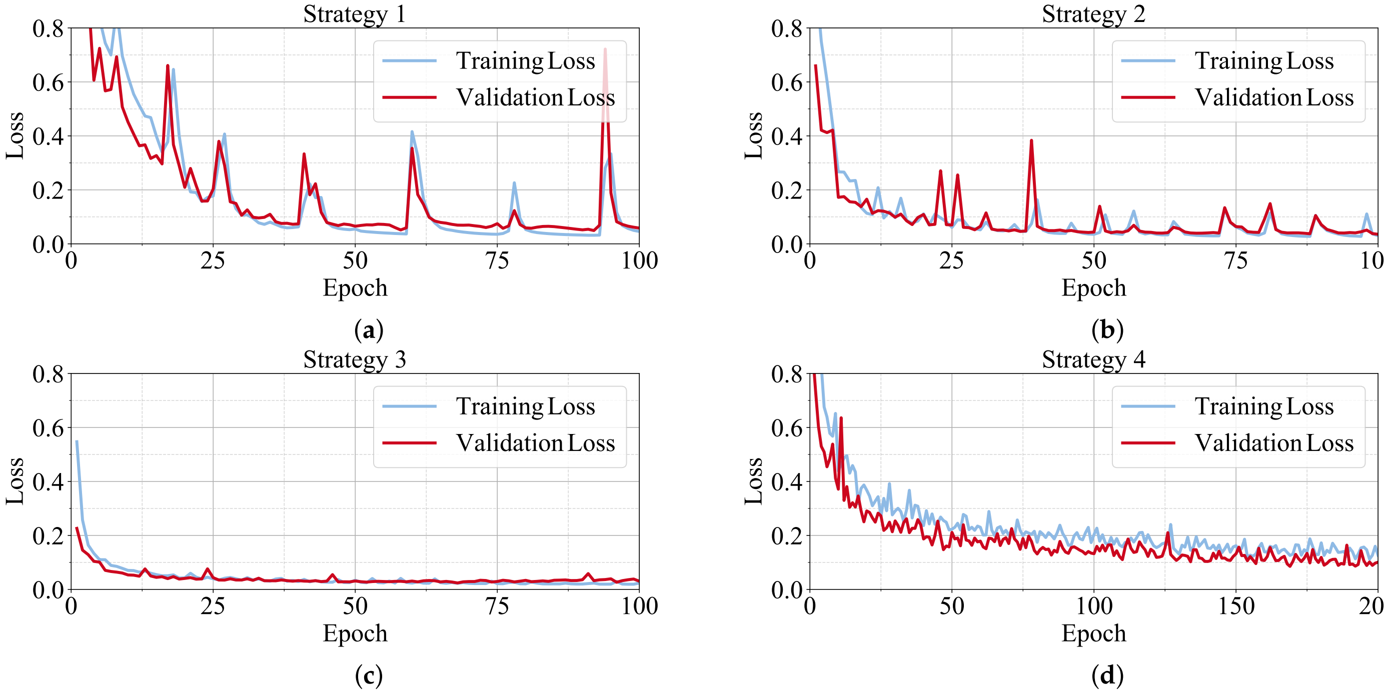
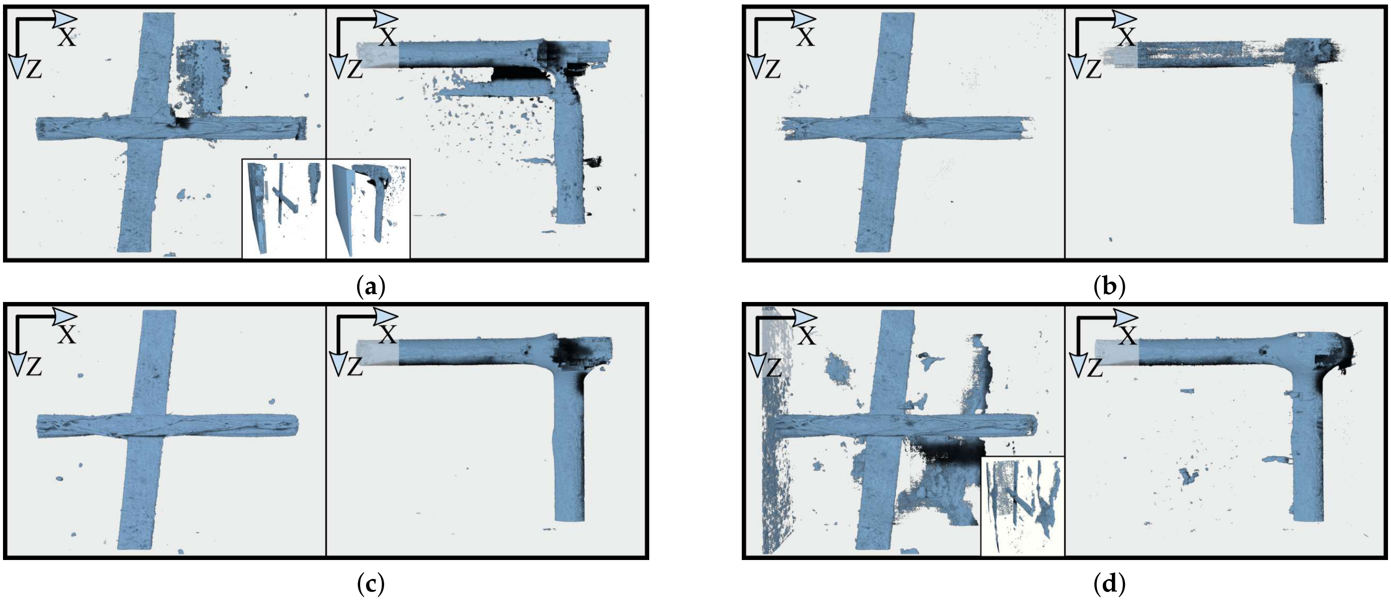


Figure 19. Training and validation loss for all four strategies. (**a**) Strategy 1: No augmentation. (**b**) Strategy 2: Weak offline augmentation. (**c**) Strategy 3: Strong offline augmentation. (**d**) Strategy 4: Online augmentation.

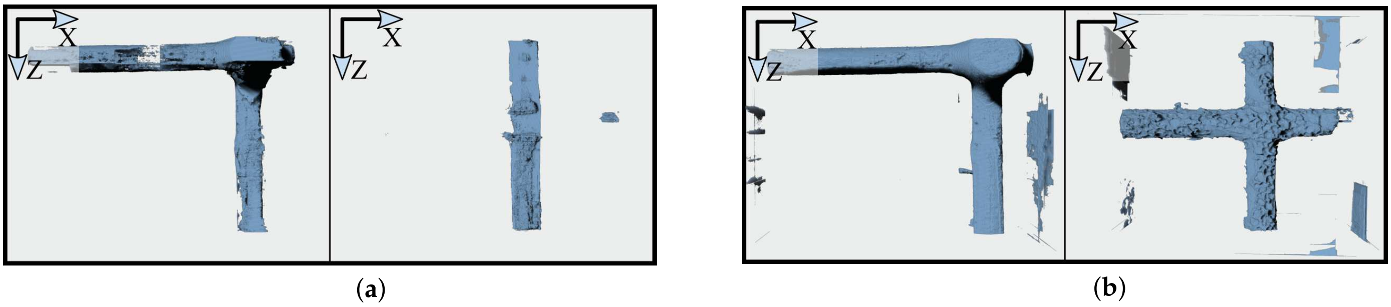
For the reasons mentioned above, the CNN trained on Strategies 1 and 2 may not perform well on Grid 2 at all. For Strategies 3 and 4, it remains unclear which CNN performs best on different grids, although Strategy 3 may have overfitted on variants of Grid 1 as well. For comparison, all networks were used to segment both Sample A and Sample C, as visualized in Figure 20. Except for the CNN without data augmentation (Figure 20a), all networks were able to preserve the shape of the roving of Sample A. However, the CNN trained with Strategy 2 failed on Grid 2 (Figure 20b). The 3D U-Net models trained with Strategies 3 and 4 were able to segment Grid 2 (Figure 20c,d), with Strategy 4 slightly outperforming Strategy 3 in the overlap region of the rovings (Figure 20d right). The heavily augmented dataset resulted in fewer artifacts (Figure 20c), although there were still gaps in the segmentation within the overlap region of the rovings in Sample C. For Sample B, all networks performed well. Sample D could only be segmented using the online augmentation approach. However, the segmentation contained many errors and was not usable. These results disproved the assumption that Strategy 3 also strongly overfitted to the Grid 1 variants but confirmed that Strategy 4 increased the variance of the datasets the most.

From the observations, both CNNs (Strategies 3 and 4) failed only in regions where they "saw" only voxels related to the roving and air. This suggests a potential solution: by resizing the volumes or scanning with larger voxel sizes, the CNN will be able to segment Grid 2 as well. This hypothesis was tested on Samples C and D, with the results visualized in Figure 21. The reconstructions were resized to a voxel size of 18.8 µm (scale factor of 0.5). The CNN trained with Strategy 3 exhibited significant performance degradation when confronted with resizing. This discrepancy can be explained by the fact that the offline augmented dataset could not achieve the same variability as that created by the online augmentation, as mentioned above. The online augmentation allowed the CNN to learn the underlying features more effectively so that it could segment the roving at larger voxel sizes. However, the CNN cannot distinguish the coating from the roving, and thus the extracted rovings were too large (see Figure 18b).



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

Figure 20. Segmentation of Sample A (left in all subimages) and Sample C (right in all subimages) using each of the four strategies. (**a**) Strategy 1: No augmentation with removed artifacts in frontal views and full segmentation as miniatures. (**b**) Strategy 2: Weak offline augmentation. (**c**) Strategy 3: Strong offline augmentation. (**d**) Strategy 4: Online augmentation with removed artifacts in Sample A (frontal view) and full segmentation as miniatures.



(**a**) (**b**)

Figure 21. Test of Strategies 3 and 4 on resized versions of Samples C and D (scale factor: 0.5, voxel size: 18.8 µm). (**a**) Strategy 3 on resized Samples C (**left**) and D (**right**). (**b**) Strategy 4 on resized Samples C (**left**) and D (**right**).

For Grid 1, Strategy 3 produced the best visual segmentation, followed by the online augmentation approach, despite the presence of additional artifacts (Figure 20d, left). However, the remaining noise is eliminated during post-processing in the roving extraction phase. With respect to Grid 2 and the general applicability, online augmentation proved to be the most effective strategy. Therefore, the 3D U-Net with online augmentation was preferred for the following steps, as it performed best overall.

3.3. Roving Extraction

The 3D U-Net trained with Strategy 4 was used to extract all rovings, as it visually performed best in the roving segmentation task. Although this strategy contains many artifacts, the Roving Surface Extractor (link in Data Availability Statement) was able to remove them, and only the rovings (and coating) were obtained (Figure 22). The rovings of Samples A, B, and C were extracted almost perfectly, while in Sample D, a small fraction (≈3 mm) is missing.

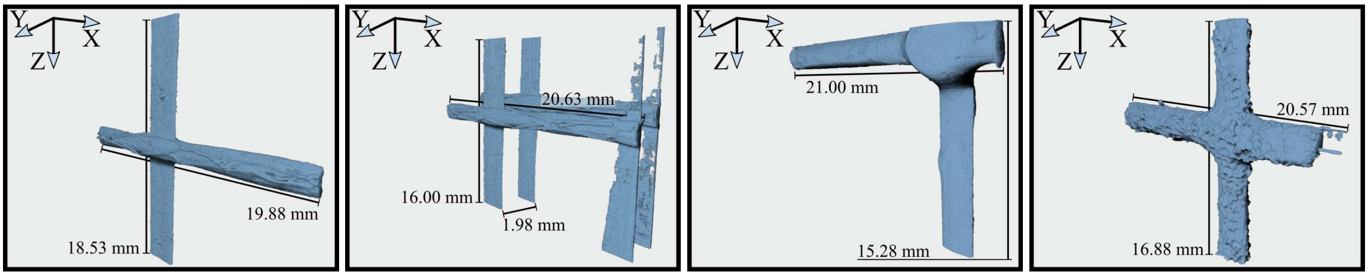


Figure 22. Post-processed rovings using the 3D U-Net trained with Strategy 4. From left to right: Samples A, B, C, and D.

3.4. Parameterized RVE and Definition of Characteristic Geometric Properties

Of the four extracted samples, Samples C and D contained too much coating, resulting in incorrect roving dimensions; see Figure 18. Sample B contained two layers of textile and two roving intersections and was therefore not suitable as an introductory example. Therefore, the characteristic properties for Sample A were derived.

Figure 23a illustrates the sample size and the size of the RVE that was modeled. Figure 23b shows Sample A. Figure 23c depicts the point cloud of the extracted roving. Finally, Figure 23d shows the obtained surface model of the roving that will be embedded in the RVE. For an illustration of the complete modeling process from CT reconstruction to parameterized RVE, refer to Figure 2.

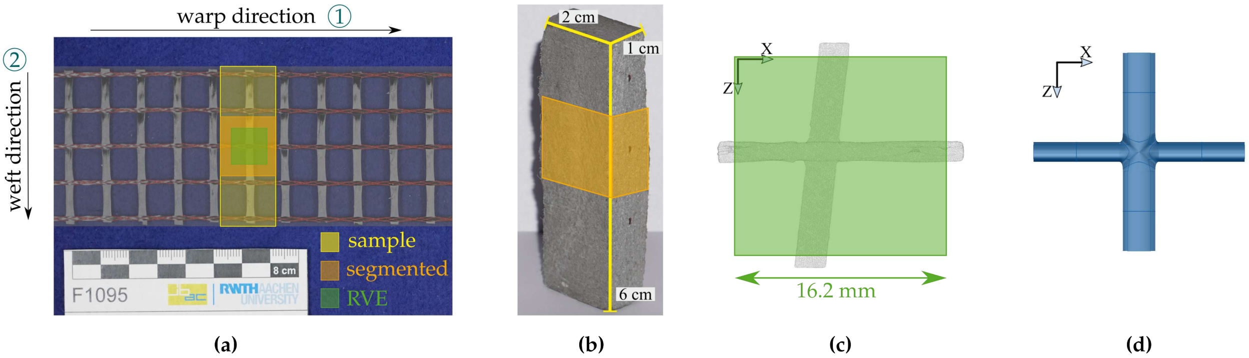


Figure 23. (**a**) Textile of Sample A indicating the sample, segmented size, and size of representative volume element (RVE). (**b**) Sample A with highlighted scanned region. (**c**) Point cloud representing the roving in (**a**), RVE size indicated. (**d**) Surface model of the roving from the parameterized RVE.

The segmented area had the dimensions of 19.88 mm (warp) × 18.53 mm (weft). Further, it can also be seen that both rovings were not orthogonal to each other; see Figure 23c. The roving in the weft direction was not parallel to the z-axis but rotated by approximately 6.325°. The voxel size of Sample A was 9.5 µm.

3.4.1. Roving Dimensions b_1/b_2 and h_1/h_2

To approximate the roving dimensions of the textile in Sample A, the method described in Section 2.7.1 was used for the x- and z-directions, respectively. The number of n_{slice} slices used for the evaluation is crucial.

For each slice, the concave hull algorithm was used to derive a polygon representing the area described by the point cloud. The area of the polygon corresponds to the cross-sectional area of the roving for the respective slice. The analysis was carried out for a varying number of slices, and the results for the obtained averaged cross-sectional areas are depicted in Figure 24. The reference solution (plotted as a dashed line) was obtained with $n_{slice} = 100$. It was observed that, for an increasing number of slices, the averaged cross-sectional area $\hat{A}_{roving}$ converged to a fixed value.

In the following, $n_{slice} = 40$ was chosen to derive the roving geometry. Applying the concave hull algorithm to the point cloud of the extracted roving and using Equation (4), the averaged properties for each roving (in x- and z-directions) were obtained. Following Equation (5), the dimensions of the ellipse approximating the rovings were derived and are summarized in Table 4.

Table 4. Derived cross-sectional dimensions of cylindrical approximation of roving using $n_{slice} = 40$.

	A in mm^2	2b in mm	2h in mm	$\kappa = h/b$
warp direction	0.83	1.49	0.71	0.48
weft direction	0.71	2.45	0.37	0.15

Since the roving in the weft direction was not perfectly parallel to the z-axis (see Figure 2c), the derived dimensions may be inaccurate. To investigate this issue, the extracted surface was rotated by 6.325° around the y-axis. Again, the width, height, and cross-sectional areas of the rovings in both directions were obtained. The maximum difference amounts to 1.1% for the width of the roving (weft direction). In the civil engineering context, this is considered negligible.

The textile used in Sample A was SITgrid 044 VL with yarn spacings of 16.20 mm (warp) × 14.70 mm (weft) (Section 2.1). According to the manufacturer, the cross-sectional area of the textile is 35.25 mm^2/m. From this, cross-sectional areas of 0.57 mm^2 (warp) and 0.52 mm^2 (weft) were derived, which is a significant difference compared to the results in Table 4. The deviation can be explained by the coating of the rovings and the knitting thread (shown in red in Figure 4a). Because the training dataset also contained the knitting thread as ground truth, and since the grayscale value and structure of the knitting thread and coating are similar to those of the roving, the segmentation could not distinguish between them. Figure 17b illustrates the problem. The knitting thread runs along the x-axis below the roving and is repeatedly interrupted, but the grayscale value is indistinguishable from that of the roving itself. The same applies to the coating.

However, since the textile was fully impregnated and the impregnation and textile were fully bonded, we used the values obtained from computed tomography data for further calculations.

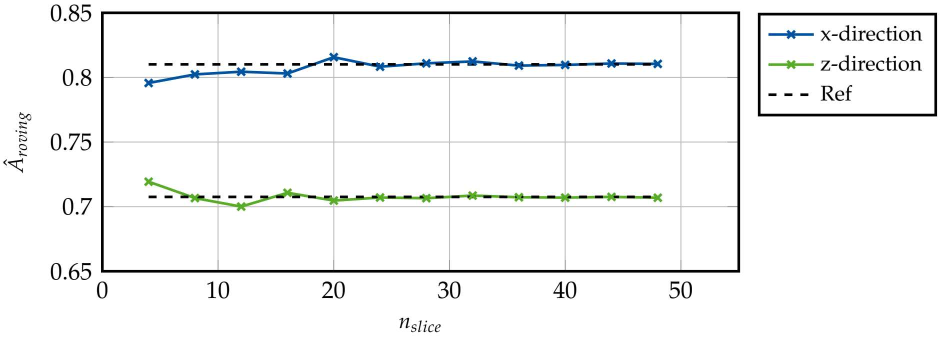


Figure 24. Obtained cross-sectional area of roving $\hat{A}_{roving}$ for varying number of slices n_{slice} for each direction of Sample A.

3.4.2. In-Plane Dimensions l_1/l_2

The in-plane dimensions depend on the textile and can be obtained from the manufacturer's data sheet. It was taken as the distance between the center lines of the rovings in the warp and weft directions. For the present example (Sample A), the dimensions were

$$l_1 = 16.20\,\text{mm} \quad \text{and} \quad l_2 = 14.70\,\text{mm},$$

as mentioned in Section 2.1. The in-plane RVE size, in comparison to the sample size and the segmented area, is shown in Figure 3a.

3.4.3. Shell Thickness h_{RVE}

The sample was produced using the Laboratory Mortar Extruder (LabMorTex) [5]. The employed mouthpiece had an opening height of $h = 10\,\text{mm}$; however, due to the extrusion process, the thickness of the extruded components was larger than the mouthpiece opening.

Using 2D binary images describing the sample at 35 locations along the scanning direction (z-axis), an averaged shell thickness was obtained using the procedure described in Section 2.7.3. Again, $n_{slice} = 40$ was chosen; however, five images were discarded since they depicted the intersection of the two rovings.

For the present example, the shell thickness derived from the computed tomography images was obtained as $h = h_{RVE} = 13$ mm. The deviation between the expected height, corresponding to the mouthpiece opening, and the derived thickness from computed tomography originates from the extrusion process and is discussed in more detail in Kalthoff et al. [5].

3.4.4. Concrete Cover c_0

The concrete cover c_0 depends on the sample. Here, the textile was positioned at the center of the component [5]. Especially for thin structures, the position of the textile within the shell is an important characteristic, which is why a verification using computed tomography images is useful.

Using the same 35 locations along the scan direction that have been used for the determination of the shell thickness, the average concrete cover c_0 is derived using the method from Section 2.7.4 and is obtained as $c_0 = 5.92\,\text{mm}$.

For validation, the theoretical concrete cover was calculated using the derived roving dimensions and shell thickness for the case of a perfectly centrally aligned roving as:

$$c_{0,ref} = \frac{h_{RVE} - (2h_2 + h_1)}{2} = 5.55\,\text{mm}. \tag{6}$$

It can be concluded that the textile was successfully placed at the center of the component during the production process, since $c_0 \approx c_{0,ref}$.

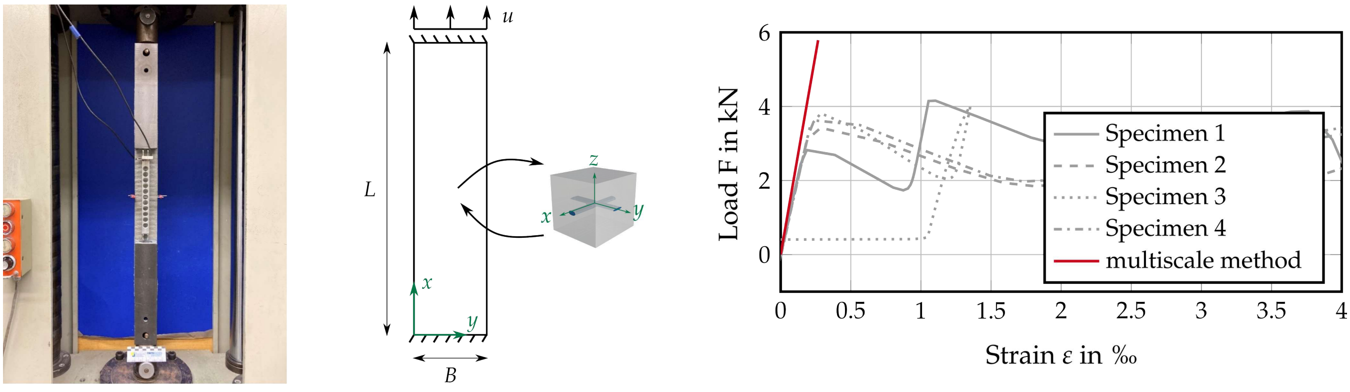
3.4.5. Example

Sample A was taken from an extruded concrete specimen, which was experimentally tested in a tensile test [5]. The setup is depicted in Figure 25a. The free length is given by $L = 250$ mm, and the width of the specimen is $B = 60$ mm. These dimensions characterize the macroscopic shell; see Figure 25b (left). The example has been presented in [65]. In contrast, here the geometric properties of the RVE were derived according to Sections 3.4.1–3.4.4. The constructed representative volume element was embedded in a multiscale framework; see Figure 25b. The material parameters are given in Table 5 and were chosen according to [5]. Here, for an initial comparison, only linear-elastic material behavior was assumed. Following the experimental test, the analysis was conducted using displacement control, where $u = 2$ mm/m. The 3D model of the RVE used for analysis is published on Zenodo [63].

Table 5. Material parameters for tension test.

	Roving	Concrete
Young's modulus E in N/mm^2	142,000	27,000
Poisson's ratio ν	0.35	0.2

Figure 25c shows the load-strain curve for four tested extruded specimens and the proposed multiscale approach. Good agreement in the linear-elastic regime can be observed. This proves the applicability of the multiscale approach for the analysis of textile-reinforced concrete structures. Furthermore, it is shown that the derived geometric properties yield the correct volume fractions between textile and concrete. In the future, more examples need to be considered that take the positioning of the textile within the concrete into account, e.g., in bending-dominated examples. The non-linear effects, such as degradation of the concrete and debonding between roving and concrete, need to be considered in future work. Kikis et al. [43] have presented a microplane-damage model in the context of a shell homogenization approach, which is able to represent the nonlinear material behavior of concrete. Here, the material parameters have been taken from [5]; in the future, these should be calibrated and validated by means of multiple examples. The use of a more advanced material model will then allow for evaluation of the load-bearing behavior of textile-reinforced concrete structures.



(a) **(b)** **(c)**

Figure 25. (**a**) Experimental setup. (**b**) Adaptation for multiscale approach. (**c**) Load-strain curve for tensile test. (Adapted with permission from Ref. [65]. 2023, L. Mester).

3.4.6. Assumptions

Finally, the assumptions made in Section 2.7.5 are evaluated with regard to their applicability to the selected example.

1. Orthogonality: From Figure 2c, it is clear that the rovings in warp and weft directions were not perfectly orthogonal to each other. However, the rotation of the extracted roving led to only small deviations in the derived geometric properties. Additionally, the distortion was introduced by the extrusion process. Using classical production methods, the distortion of the textile is less distinct. Thus, the assumption of orthogonal rovings is valid.
2. A single roving intersection is sufficient for the RVE. This assumption is valid as long as manufacturing errors, which could lead to varying shell thicknesses or concrete covers, can be excluded. Ideally, the segmented area is greater than or equal to the RVE size in order to properly approximate the roving dimensions.
3. Elliptical cross-sections are assumed for both rovings. Analysis of the cross-sectional images revealed that small height-to-width ratios $\kappa = h/b$ should be considered for the roving. In the context of the presented linear-elastic tensile test, the shape approximation proved to be satisfactory.

For the selected example, the assumptions made are valid. In general, the assumption of the rovings being orthogonal to each other only holds as long as the warp and weft direction of the textile are orthogonal. For other textile geometries, the parameterized RVE has to be adapted. Note that the employed periodic boundary conditions in the multiscale approach require the RVE to be symmetric and point symmetric. Thus, a non-symmetric

RVE requires different boundary conditions for the homogenization process. A possible solution is discussed in [65].

The investigation of only a single roving intersection is sufficient as long as it is periodically repeating and representative of the whole structure. If the reinforcement layout changes within the structure, e.g., multiple layers of textile in highly stressed areas, multiple RVEs have to be defined. These can be used for homogenization of the corresponding areas.

The assumption of elliptical roving geometries may not be sufficient if debonding or friction are to be investigated. That would require a different algorithm to define the roving geometries; one possibility is presented in Zhang et al. [66] for modeling arteries. Due to the limitations imposed by the scaling center, e.g., star-shapedness, automation is not trivial. Again, the chosen homogenization process requires symmetry and point symmetry, which further complicates the possible description of an RVE.

4. Conclusions

A multiscale method for the analysis of thin structures enhanced by image-based methods was presented. It can be used for the evaluation of novel construction methods and for the structural analysis of textile-reinforced concrete shells.

The entire process, ranging from the generation of computed tomography data to the segmentation and derivation of characteristic geometric properties for the multiscale method, was demonstrated on an extruded concrete specimen. The computed tomography images were processed using a 3D convolutional neural network to obtain a volumetric reconstruction of the grid-aligned rovings within the mortar matrix. Four different augmentation strategies were presented, and the efficiency of the trained CNNs was compared. The derived grid properties were validated by comparison with values from existing literature. A linear-elastic benchmark example showed the validity of the multiscale approach.

However, there is still room for improvement.

In terms of roving extraction, the CNN training dataset can be extended with different textiles and/or the implementation of an unpaired volume-to-volume translation for the training data generation based on [67].

For the structural analysis of concrete specimens, further validation using more complex examples is required to validate the derived geometric properties. For the analysis of specimens with multiple reinforcement layers, the proposed parameterized RVE can be used multiple times on top of one another, so that an extension is straightforward. The analysis of specimens where the weft and warp directions of the employed textile are not orthogonal to each other requires the definition of an adapted parameterized RVE and, consequently, may require different boundary conditions within the multiscale framework. To evaluate the non-linear material behavior of concrete, a microplane damage model can be used [43]. The calibration and validation of the respective material parameters using several different experimental examples is part of future work. At this stage, a full bond between concrete and roving is assumed. However, when investigating debonding behavior, the assumption of an elliptical shape of the roving may not be sufficient. For these cases, it may be beneficial to incorporate the exact roving geometry in the RVE. However, this cannot necessarily be incorporated directly into the multiscale framework.

Author Contributions: Conceptualization, F.W. and L.M.; methodology, F.W. and L.M.; software, F.W. and L.M.; validation, F.W. and L.M.; formal analysis, F.W. and L.M.; investigation, F.W. and L.M.; resources, H.-G.M. and S.K.; data curation, F.W. and L.M.; writing—original draft preparation, F.W., L.M., H.-G.M. and S.K.; writing—review and editing, F.W., L.M., H.-G.M. and S.K.; visualization, F.W. and L.M.; supervision, H.-G.M. and S.K.; project administration, H.-G.M. and S.K.; funding acquisition, H.-G.M. and S.K. All authors have read and agreed to the published version of the manuscript.

Funding: The research work presented in the publication has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—SFB/TRR 280; Project-ID: 417002380.

Data Availability Statement: Software available at: https://gitlab.com/fra-wa/aiseg (AiSeg for CNN training, accessed on 17 September 2023); https://gitlab.com/fra-wa/roving_surface_extractor (Roving Surface Extractor, accessed on 17 September 2023). CNN training data available at: https://doi.org/10.34740/KAGGLE/DS/2920892 (Roving Segmentation Dataset, accessed on 17 September 2023). Parameterized RVE model available at: https://doi.org/10.5281/zenodo.8340828 (accessed on 15 September 2023).

Acknowledgments: The authors are grateful to the Center for Information Services and High Performance Computing [Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)] at TUD Dresden University of Technology for providing its facilities for high throughput calculations.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional neural network
CRC	Carbon reinforced concrete
CT	Computed tomography
LabMorTex	Laboratory Mortar Extruder
NURBS	Non-uniform rational B-splines
RVE	Representative volume element
SBIGA	Scaled boundary isogeometric analysis

References

1. Scheerer, S.; Zobel, R.; Müller, E.; Senckpiel-Peters, T.; Schmidt, A.; Curbach, M. Flexural Strengthening of RC Structures with TRC—Experimental Observations, Design Approach and Application. *Appl. Sci.* **2019**, *9*, 1322 . [CrossRef]
2. Beckmann, B.; Bielak, J.; Bosbach, S.; Scheerer, S.; Schmidt, C.; Hegger, J.; Curbach, M. Collaborative research on carbon reinforced concrete structures in the CRC/TRR 280 project. *Civ. Eng. Des.* **2021**, *3*, 99–109. [CrossRef]
3. Bielak, J.; Kollegger, J.; Hegger, J. Shear in Slabs with Non-Metallic Reinforcement. Ph.D. Thesis, Lehrstuhl und Institut für Massivbau, RWTH Aachen University, Aachen, Germany 2021. [CrossRef]
4. Morales Cruz, C. Supplementary Data to Crack-distributing Carbon Textile Reinforced Concrete Protection Layers, Ph.D. Thesis, Lehrstuhl für Baustoffkunde–Bauwerkserhaltung, RWTH Aachen University, Aachen, Germany, 2022. [CrossRef]
5. Kalthoff, M.; Raupach, M.; Matschei, T. Investigation into the Integration of Impregnated Glass and Carbon Textiles in a Laboratory Mortar Extruder (LabMorTex). *Materials* **2021**, *14*, 7406. [CrossRef]
6. Kalthoff, M.; Raupach, M.; Matschei, T. Extrusion and Subsequent Transformation of Textile-Reinforced Mortar Components—Requirements on the Textile, Mortar and Process Parameters with a Laboratory Mortar Extruder (LabMorTex). *Buildings* **2022**, *12*, 726. [CrossRef]
7. Kalthoff, M.; Bosbach, S.; Backes, J.G.; Morales Cruz, C.; Claßen, M.; Traverso, M.; Raupach, M.; Matschei, T. Fabrication of lightweight, carbon textile reinforced concrete components with internally nested lattice structure using 2-layer extrusion by LabMorTex. *Constr. Build. Mater.* **2023**, *395*, 132334. [CrossRef]
8. Mester, L.; Wagner, F.; Liebold, F.; Klarmann, S.; Maas, H.G.; Klinkel, S. Image-Based Modelling and Analysis of Carbon-Fibre Reinforced Concrete Shell Structures. In Proceedings of the Concrete Innovation for Sustainability, Oslo, Norway, 12–16 June 2022; Volume 6, pp. 1631–1640.
9. Scholzen, A.; Chudoba, R.; Hegger, J. Dünnwandiges Schalentragwerk aus textilbewehrtem Beton. *Beton- Und Stahlbetonbau* **2012**, *107*, 767–776. [CrossRef]
10. Wagner, F.; Eltner, A.; Maas, H.G. River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *119*, 103305. [CrossRef]
11. Wagner, F. Carbon Rovings Segmentation Dataset. 2023. Available online: https://www.kaggle.com/datasets/franzwagner/carbon-rovings (accessed on 22 August 2023).
12. Berger, M.; Tagliasacchi, A.; Seversky, L.M.; Alliez, P.; Guennebaud, G.; Levine, J.A.; Sharf, A.; Silva, C.T. A Survey of Surface Reconstruction from Point Clouds. *Comput. Graph. Forum* **2017**, *36*, 301–329. [CrossRef]

13. Huang, Z.; Wen, Y.; Wang, Z.; Ren, J.; Jia, K. Surface Reconstruction from Point Clouds: A Survey and a Benchmark. *arXiv* **2022**, arXiv:2205.02413. https://doi.org/10.48550/arXiv.2205.02413.
14. Vukicevic, A.M.; Çimen, S.; Jagic, N.; Jovicic, G.; Frangi, A.F.; Filipovic, N. Three-dimensional reconstruction and NURBS-based structured meshing of coronary arteries from the conventional X-ray angiography projection images. *Sci. Rep.* **2018**, *8*, 1711. [CrossRef]
15. Wang, Y.; Gao, L.; Qu, J.; Xia, Z.; Deng, X. Isogeometric analysis based on geometric reconstruction models. *Front. Mech. Eng.* **2021**, *16*, 782–797. [CrossRef]
16. Grove, O.; Rajab, K.; Piegl, L.A. From CT to NURBS: Contour Fitting with B-spline Curves. *Comput.-Aided Des. Appl.* **2010**, *7*, 1–19. [CrossRef]
17. Chasapi, M.; Mester, L.; Simeon, B.; Klinkel, S. Isogeometric analysis of 3D solids in boundary representation for problems in nonlinear solid mechanics and structural dynamics. *Int. J. Numer. Methods Eng.* **2021**. [CrossRef]
18. Hughes, T.J.R.; Cottrell, J.A.; Bazilevs, Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 4135–4195. [CrossRef]
19. Cottrell, J.A.; Hughes, T.J.R.; Bazilevs, Y. *Isogeometric Analysis: Toward Integration of CAD and FEA*; John Wiley and Sons: Hoboken, NJ, USA , 2009.
20. ProCon X-Ray GmbH. PXR PROCON X-RAY. 2020. Available online: https://procon-x-ray.de/ct-xpress/ (accessed on 17 September 2023).
21. Millner, M.R.; Payne, W.H.; Waggener, R.G.; McDavid, W.D.; Dennis, M.J.; Sank, V.J. Determination of effective energies in CT calibration. *Med Phys.* **1978**, *5*, 543–545. [CrossRef]
22. Tan, Y.; Kiekens, K.; Welkenhuyzen, F.; Kruth, J.; Dewulf, W. Beam hardening correction and its influence on the measurement accuracy and repeatability for CT dimensional metrology applications. In Proceedings of the 4th Conference on Industrial Computed Tomography (iCT), Wels, Austria, 19–21 September 2012; Volume 17, pp. 355–362.
23. Ciresan, D.; Giusti, A.; Gambardella, L.; Schmidhuber, J. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. *Adv. Neural Inf. Processing Syst.* **2012**, 25, 1–9. Available online: https://papers.nips.cc/paper_files/paper/2012/hash/459a4ddcb586f24efd9395aa7662bc7c-Abstract.html (accessed on 22 August 2023).
24. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Munich, Germany, 5–9 October 2015; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 234–241. [CrossRef]
25. Badrinarayanan, V.; Handa, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling. *arXiv* **2015**, arXiv:1505.07293. https://doi.org/10.48550/arXiv.1505.07293.
26. Xiao, T.; Liu, Y.; Zhou, B.; Jiang, Y.; Sun, J. Unified Perceptual Parsing for Scene Understanding. In Proceedings of the Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 432–448. [CrossRef]
27. Peng, C.; Zhang, X.; Yu, G.; Luo, G.; Sun, J. Large Kernel Matters—Improve Semantic Segmentation by Global Convolutional Network. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Los Alamitos, CA, USA, 2017; pp. 1743–1751. [CrossRef]
28. Yu, L.; Cheng, J.Z.; Dou, Q.; Yang, X.; Chen, H.; Qin, J.; Heng, P.A. Automatic 3D Cardiovascular MR Segmentation with Densely-Connected Volumetric ConvNets. *Lect. Notes Comput. Sci. (LCNS)* **2017**, *10434*, 287–295. [CrossRef]
29. Bui, T.D.; Shin, J.; Moon, T. Skip-connected 3D DenseNet for volumetric infant brain MRI segmentation. *Biomed. Signal Process. Control.* **2019**, *54*, 101613. [CrossRef]
30. Chen, S.; Ma, K.; Zheng, Y. Med3D: Transfer Learning for 3D Medical Image Analysis. *arXiv* **2019**, arXiv:1904.00625. https://doi.org/10.48550/arXiv.1904.00625.
31. Çiçek, Ö.; Abdulkadir, A.; Lienkamp, S.S.; Brox, T.; Ronneberger, O. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Proceedings of the Medical Image Computing and Computer-Assisted Intervention—MICCAI 2016, Athens, Greece, 17–21 October 2016*; Ourselin, S., Joskowicz, L., Sabuncu, M.R., Unal, G., Wells, W., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 424–432. [CrossRef]
32. Wagner, F.; Maas, H.G. A Comparative Study of Deep Architectures for Voxel Segmentation in Volume Images. In Proceedings of the ISPRS Geospatial Week 2023, Cairo, Egypt, 2–7 September 2023.
33. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
34. Buslaev, A.; Iglovikov, V.I.; Khvedchenya, E.; Parinov, A.; Druzhinin, M.; Kalinin, A.A. Albumentations: Fast and Flexible Image Augmentations. *Information* **2020**, *11*, 125. [CrossRef]
35. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Neural Information Processing Systems (NIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; pp. 8026–8037. Available online: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf (accessed on 22 August 2023).
36. Alam, M.; Wang, J.F.; Guangpei, C.; Yunrong, L.V.; Chen, Y. Convolutional Neural Network for the Semantic Segmentation of Remote Sensing Images. *Mob. Netw. Appl.* **2021**, *26*, 200–215. [CrossRef]

37. Xu, H.; He, H.; Zhang, Y.; Ma, L.; Li, J. A comparative study of loss functions for road segmentation in remotely sensed road datasets. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *116*, 103159. [CrossRef]
38. Wu, Y.; He, K. Group Normalization. *Int. J. Comput. Vis.* **2020**, *128*, 742–755. [CrossRef]
39. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015; Conference Track Proceedings; Bengio, Y., LeCun, Y., Eds.; Cornel University Press: Ithaca, NY, USA, 2017. . [CrossRef]
40. Taha, A.A.; Hanbury, A. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. *BMC Med. Imaging* **2015**, *15*, 29. [CrossRef]
41. Cardoso, M.J.; Li, W.; Brown, R.; Ma, N.; Kerfoot, E.; Wang, Y.; Murrey, B.; Myronenko, A.; Zhao, C.; Yang, D.; et al. MONAI: An open-source framework for deep learning in healthcare. *arXiv* **2022**, arXiv:2211.02701. https://doi.org/10.48550/arXiv.2211.02701.
42. Lorensen, W.E.; Cline, H.E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the ACM SIGGRAPH Computer Graphics*; Association for Computing Machinery: New York, NY, USA, 1987; Volume 21, SIGGRAPH '87; pp. 163–169. . [CrossRef]
43. Kikis, G.; Mester, L.; Spartali, H.; Chudoba, R.; Klinkel, S. Analyse des Trag- und Bruchverhaltens von Carbonbetonstrukturen im Rahmen des SFB/TRR 280/Analysis of the load-bearing and fracture behavior of carbon concrete structures as part of the SFB/TRR 280. *Bauingenieur* **2023**, *98*, 218–226. [CrossRef]
44. Platen, J.; Zreid, I.; Kaliske, M. A nonlocal microplane approach to model textile reinforced concrete at finite deformations. *Int. J. Solids Struct.* **2023**, *267*, 112151. [CrossRef]
45. Valeri, P.; Fernàndez Ruiz, M.; Muttoni, A. Tensile response of textile reinforced concrete. *Constr. Build. Mater.* **2020**, *258*, 119517. [CrossRef]
46. Schröder, J. A numerical two-scale homogenization scheme: The FE2-method. In *Plasticity and Beyond: Microstructures, Crystal-Plasticity and Phase Transitions*; Springer: Vienna, Austria, 2014; pp. 1–64. [CrossRef]
47. Saeb, S.; Steinmann, P.; Javili, A. Aspects of Computational Homogenization at Finite Deformations: A Unifying Review From Reuss' to Voigt's Bound. *Appl. Mech. Rev.* **2016**, *68*, 050801. [CrossRef]
48. Coenen, E.; Kouznetsova, V.; Geers, M. Computational homogenization for heterogeneous thin sheets. *Int. J. Numer. Methods Eng.* **2010**, *83*, 1180–1205. [CrossRef]
49. Hii, A.K.; El Said, B. A kinematically consistent second-order computational homogenisation framework for thick shell models. *Comput. Methods Appl. Mech. Eng.* **2022**, *398*, 115136. [CrossRef]
50. Börjesson, E.; Larsson, F.; Runesson, K.; Remmers, J.J.; Fagerström, M. Variationally consistent homogenisation of plates. *Comput. Methods Appl. Mech. Eng.* **2023**, *413*, 116094. [CrossRef]
51. Gruttmann, F.; Wagner, W. A coupled two-scale shell model with applications to layered structures. *Int. J. Numer. Methods Eng.* **2013**, *94*, 1233–1254. [CrossRef]
52. Feyel, F.; Chaboche, J.L. FE2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials. *Comput. Methods Appl. Mech. Eng.* **2000**, *183*, 309–330. [CrossRef]
53. Hill, R. Elastic properties of reinforced solids: Some theoretical principles. *J. Mech. Phys. Solids* **1963**, *11*, 357–372. [CrossRef]
54. Miehe, C.; Koch, A. Computational micro-to-macro transitions of discretized microstructures undergoing small strains. *Arch. Appl. Mech. Ingenieur Arch.* **2002**, *72*, 300–317. [CrossRef]
55. Mester, L.; Klarmann, S.; Klinkel, S. Homogenization assumptions for the two-scale analysis of first-order shear deformable shells. *Comput. Mech.* **2023**. [CrossRef]
56. Song, C. *The Scaled Boundary Finite Element Method*; John Wiley & Sons, Ltd.: Chichester, UK, 2018. [CrossRef]
57. Jüttler, B.; Maroscheck, S.; Kim, M.S.; Youn Hong, Q. Arc fibrations of planar domains. *Comput. Aided Geom. Des.* **2019**, *71*, 105–118. [CrossRef]
58. Trautner, S.; Jüttler, B.; Kim, M.S. Representing planar domains by polar parameterizations with parabolic parameter lines. *Comput. Aided Geom. Des.* **2021**, *85*, 101966. [CrossRef]
59. Chin, E.B.; Sukumar, N. Scaled boundary cubature scheme for numerical integration over planar regions with affine and curved boundaries. *Comput. Methods Appl. Mech. Eng.* **2021**, *380*, 113796. [CrossRef]
60. Sauren, B.; Klarmann, S.; Kobbelt, L.; Klinkel, S. A mixed polygonal finite element formulation for nearly-incompressible finite elasticity. *Comput. Methods Appl. Mech. Eng.* **2023**, *403*, 115656. [CrossRef]
61. Reichel, R.; Klinkel, S. A non–uniform rational B–splines enhanced finite element formulation based on the scaled boundary parameterization for the analysis of heterogeneous solids. *Int. J. Numer. Methods Eng.* **2023**, *124*, 2068–2092. [CrossRef]
62. Bauer, B.; Arioli, C.; Simeon, B. Generating Star-Shaped Blocks for Scaled Boundary Multipatch IGA. In *Isogeometric Analysis and Applications 2018*; Lecture Notes in Computational Science and Engineering; van Brummelen, H., Vuik, C., Möller, M., Verhoosel, C., Simeon, B., Jüttler, B., Eds.; Springer International Publishing: Cham, Switzerland, 2021; Volume 133, pp. 1–25. [CrossRef]
63. Mester, L.; Klinkel, S. Parameterized Representative Volume Element (RVE) for Textile-Reinfoced Composites. 2023. Available online: https://zenodo.org/record/8340828 (accessed on 22 August 2023).
64. Park, J.S.; Oh, S.J. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *J. Inf. Sci. Eng.* **2012**, *28*, 587–600.

65. Mester, L.; Klarmann, S.; Klinkel, S. Homogenisation for macroscopic shell structures with application to textile–reinforced mesostructures. *PAMM* **2023**, *22*. [CrossRef]
66. Zhang, Y.; Bazilevs, Y.; Goswami, S.; Bajaj, C.L.; Hughes, T.J.R. Patient-Specific Vascular NURBS Modeling for Isogeometric Analysis of Blood Flow. *Comput. Methods Appl. Mech. Eng.* **2007**, *196*, 2943–2959. [CrossRef] [PubMed]
67. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2242–2251. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Chapter 7

Image-to-Image Translation for Semi-Supervised Semantic Segmentation

This chapter explores the potential of using Contrastive Unpaired Translation (CUT), a type of Generative Adversarial Network (GAN), to generate synthetic training data for semantic segmentation in 2D, focusing specifically on pore segmentation in Computed Tomography (CT) data. The aim is to investigate how the input data should be prepared for the GAN, in particular to examine the composition of input images for the generation of semantic segmentation datasets. Furthermore, by incorporating basic augmentation techniques discussed in Chapter 4, both semi-supervised and unsupervised end-to-end training on synthetic data using a Convolutional Neural Network (CNN) called Global Convolutional Network (GCN) [96] with a ResNeXt 50 [97] backbone are presented. This chapter sets the stage for future research in semi-supervised 3D learning methods.

7.1 Introduction

Throughout this study, the lack of data for training neural networks for semantic segmentation has been the main problem, and basic data augmentation has proven to be a valuable tool to overcome these problems. However, without post-processing, such as refining the resulting segmentation or manipulating the input data before inference, many errors are contained in the segmentation. To effectively solve these problems, neural networks need to be more generalized, requiring more paired training data consisting of input images and their corresponding pixel-wise annotations. However, the acquisition of paired data requires significant resources and labor, a common problem in deep learning that severely limits the scalability and generalizability of segmentation models.

To tackle these hurdles, researchers have explored various methods to generate synthetic datasets for semantic segmentation [59, 98]. Among these methods, the Generative Adversarial Network named CUT has emerged as a promising technique. CUT, such as its successor CycleGAN [42], is able to learn mappings between domains without paired data, facilitating unpaired image synthesis [5]. This approach reduces reliance on annotated data, thereby enhancing scalability, particularly in domains with limited annotated images. Furthermore, CUT produces high-fidelity synthetic data closely resembling real-world distributions, potentially improving model robustness and generalization.

While promising, challenges remain, notably the domain gap between synthetic and real data distributions and the preservation of fine-grained details and semantic consistency in synthesized images or volumes. [99, 100]

Despite the given challenges, Park et al. [5] have shown the ability of CUT to translate segmentation masks into realistic images, as demonstrated on the Cityscapes dataset. Compared to other methods such as CycleGAN [42], MUNIT [62], and DRIT [101], CUT showed superior image similarity metrics while being faster and having fewer parameters than others, which is why this network is the focus of further investigation. However, the generation of images using only labels as input resulted in some structural artifacts. In Figure 7.1, the arrows highlight regions where the GAN produced unrealistic output. At arrow 1, a bus is represented by a stack of cars. At arrows 2 and 3, the GAN projected unrealistic structures onto the facades.

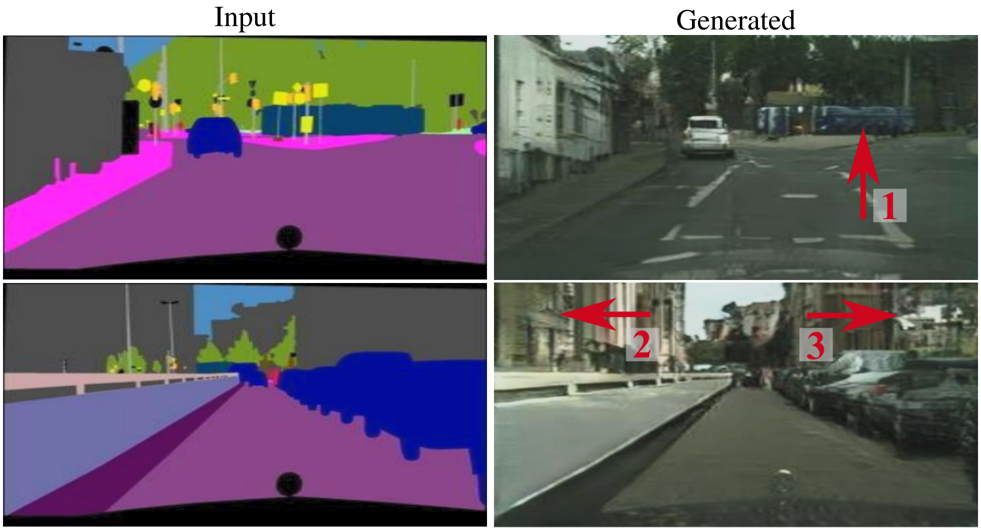


Figure 7.1: Results of the original CUT on the Cityscapes dataset [102]. Red arrows highlight image artifacts. Images adapted from Park et al. [5].

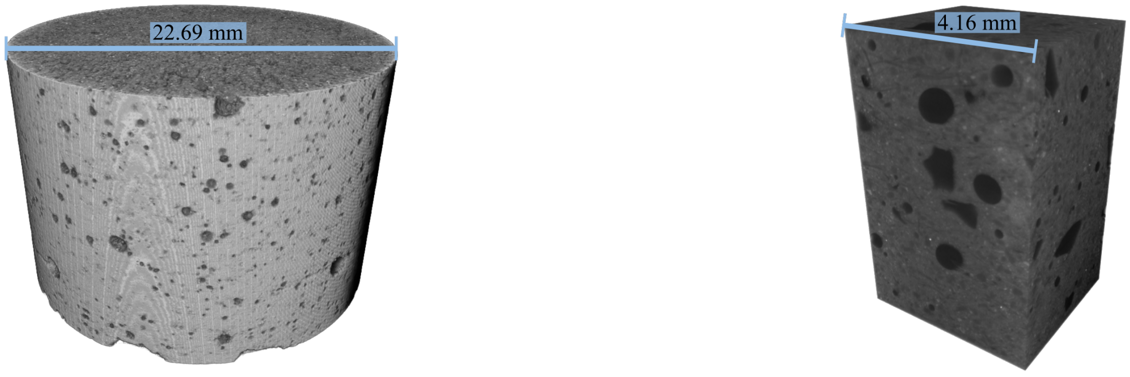
In a similar study by Imbusch et al. [63], the generation of a dataset using CUT was demonstrated. Their method generates training data for semantic segmentation without manual annotation. They use synthetic images of a rendered 3D scene, which are quite detailed, and then adapt them to the real domain using CUT to minimize the domain gap. However, no such renderer is available for CT scans of Strain-Hardening Cement-based Composite (SHCC) or Carbon Reinforced Concrete (CRC). It is therefore necessary to investigate how detailed images of the input domain need to be in order to generate synthetic training data.

Once synthetic training data is generated, an end-to-end training for semantic segmentation can be performed. In this study, the GCN with a ResNeXt 50 backbone is used, since it has a strong performance, as demonstrated on the River Water Segmentation Dataset (RIWA) dataset [2] in Chapter 4 (2nd place overall), while being more resource efficient than other networks within the top 10 on the RIWA dataset (half the size of the first place U-Net (ResNeXt 50 backbone)).

The overall goal of this investigation is to segment the pores in two CT reconstructions of different concretes without manually labeled data. The first reconstruction (Figure 7.2a) contains CT images of concrete with a rather homogeneous background. The second reconstruction is a CT scan of SHCC (Figure 7.2b). This reconstruction contains more identifiable elements due to a smaller voxel size and more intricate pores due to the water-reducing concrete process.

7.2 Methods

This section first describes GANs and the CUT architecture and why it is more efficient than for example the CycleGAN. Second, the Fréchet Inception Distance (FID), a metric to measure the similarity between two domains, is explained. Then, the datasets that are used to implement and verify the CUT model and to perform the experiments are described. Finally, the experimental design is presented, which shows the end-to-end training to segment pores using the GCN.



(a) **(b)**

Figure 7.2: 3D visualization of the CT reconstructions to be segmented. **(a)**: CT reconstruction of a concrete sample. **(b)**: CT reconstruction of a SHCC sample.

7.2.1 Generative Adversarial Networks

In a simple GAN framework, there are two main components: the generator and the discriminator. The generator tries to produce realistic data samples (e.g., images) from random noise, or images in the case of image-to-image synthesis, while the discriminator as a classification network attempts to classify the images generated by the generator into real and fake. The objective of the generator is to fool the discriminator into believing that the samples it generates are real.

As training progresses, the generator gets better at generating realistic samples that increasingly resemble real data from the training set. Conversely, the discriminator improves its ability to distinguish between real and fake samples. Through this adversarial process, both networks iteratively refine their performance. [103]

However, the generator can learn to always produce the same output, which is called mode collapse and will always fool the discriminator [104]. This is just one of many problems that have been the focus of research in recent years. In the case of unpaired image-to-image translation, for example, the goal is not only to generate realistic images, but also to preserve the shape of objects, introducing another task for a GAN to solve.

7.2.2 Contrastive Unpaired Translation

CUT efficiently solves mode collapse and image-to-image translation. Instead of more complex GANs that use a series of losses or multiple neural networks to achieve consistent translations (e.g., CycleGAN), CUT is a rather lightweight model, since it consists of only two CNNs, a generator G and a discriminator D (Figure 7.3). Furthermore, only two losses are used, the common adversarial loss and the multilayer patchwise contrastive loss, which Park et al. [5] call *PatchNCE* Loss. In the adversarial loss (Equation 7.1), the D learns to predict whether the input is real or fake, and the G tries to fool the D (Figure 7.3 Adversarial Loss).

$$L_{adversarial}(G,D,X,Y) = \mathbb{E}_{y\sim Y}\log(D(y)) + \mathbb{E}_{x\sim X}\log(1-D(G(x))) \tag{7.1}$$

where:	$L_{adversarial}$	= Adversarial Loss
	G	= Generator
	D	= Discriminator
	X	= Input domain
	Y	= Target domain
	x	= An instance of the input domain X ($\{x \in X\}$)
	y	= An instance of the target domain Y ($\{y \in Y\}$)

In *PatchNCE* loss, *NCE*, or noise contrastive estimation, is a technique for learning embeddings by contrasting positive samples with noise samples. The goal is to maximize the mutual information between patches, i.e., the head of a horse should be more closely related to the head of a zebra than to the background or a leg. This spatial input-output relationship is achieved by first taking random patches across a set of layers L of the encoding part of the generator G_{enc} (Figure 7.3 dashed, gray lines), where l is a particular layer. These patches are then fed into a small Multilayer Perceptron (MLP) network H_l (Figure 7.3 Multilayer), which outputs a stack of features $\{z_l\}_L$. Similarly, this is done with the generated image $\hat{y}$. Then, a patch at a given position s of the input feature stack should be closely related to a patch at the same position of the output feature stack at that position (Figure 7.3 teal squares and Patchwise Contrastive Learning (z^+ and z)). Furthermore, other patches within a feature stack can be used as negative samples, where the distance from the corresponding feature to the other features in the stack should be large (Figure 7.3 red squares and Patchwise Contrastive Learning (z_1^-; z_2^-; z_3^- and z^+). [5]

$$L_{PatchNCE}(G,H,X) = \mathbb{E}_{x\sim X}\sum_{l=1}^{L}\sum_{s=1}^{S_l} l(\hat{z}_l^s, z_l^s, z_l^{S\setminus s}) \tag{7.2}$$

where:	$L_{PatchNCE}$	= PatchNCE loss
	G	= Generator
	H	= MLP network
	X	= Input domain
	x	= An instance of the input domain X ($\{x \in X\}$)
	L	= Layers of interest (predefined)
	l	= A layer of G_{enc}
	s	= Spatial position within a layer l ($s \in S_l$)
	S_l	= Spatial locations S_l of a layer G_{enc}^l
	$\hat{z}_l^s$	= Feature of a generated image $\hat{y}$ at a specific spatial position s within a layer G_{enc}^l
	z_l^s	= Feature of an input image at a specific spatial position s within a layer G_{enc}^l
	$z_l^{S\setminus s}$	= Negative features within the input feature stack

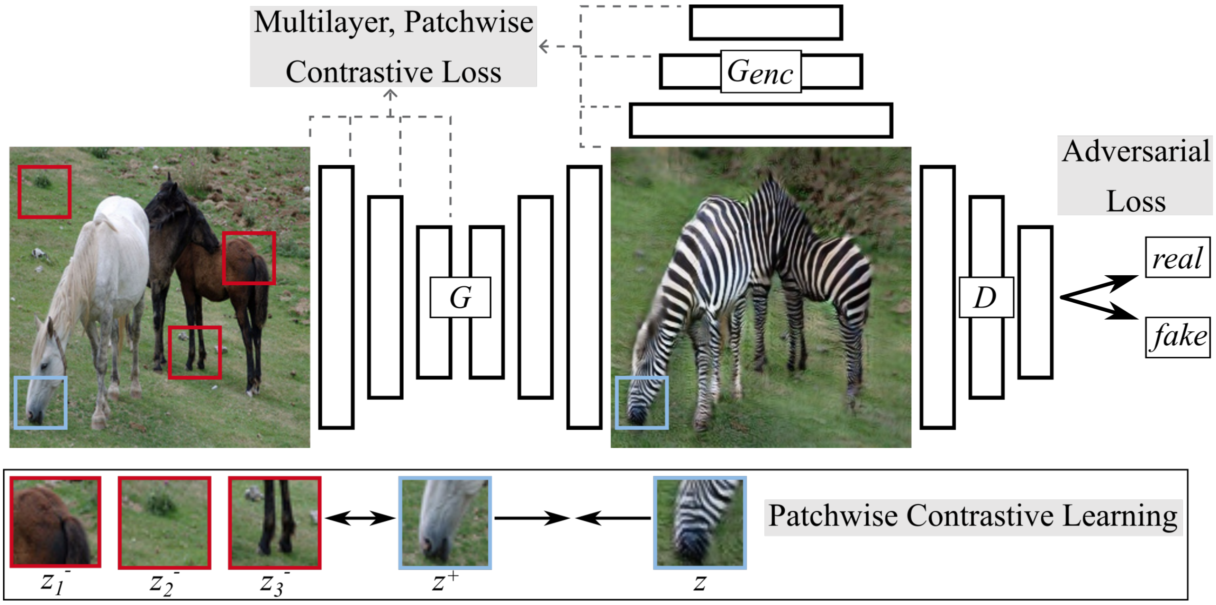

Multilayer, Patchwise Contrastive Loss
Genc
G
Adversarial Loss
D
real
fake
Patchwise Contrastive Learning
z1-
z2-
z3-
z+
z

Figure 7.3: Architecture of CUT adapted from [5]. The GAN generator (G) translates the input domain (horses) into the target domain (zebras). The Patchwise Contrastive Learning ensures that patches of the same location within the input and target domains (teal) look more similar than random patches (red). The GAN Loss of the GAN discriminator D forces the G to produce realistic images.

Finally, the *PatchNCE* loss can also be applied to the target domain Y, which helps prevent the generator from making unnecessary changes, resulting in the following final equation for the total loss:

$$L_{CUT} = L_{adversarial}(G,D,X,Y) + L_{PatchNCE}(G,H,X) + L_{PatchNCE}(G,H,Y) \tag{7.3}$$

7.2.3 Fréchet Inception Distance

The Fréchet Inception Distance (FID) serves as a measure of the similarity between two sets of images and shows a strong correlation with human judgments of visual quality. It's commonly used to evaluate the quality of samples generated by GANs [5, 42, 62, 101]. The FID is computed by determining the Fréchet distance between two Gaussian distributions fitted to feature representations extracted from the Inception network (a CNN for image classification) [105] (Equation 7.4). A lower FID score indicates greater similarity between the generated and real images, thus reflecting higher quality in the generated samples.

$$d^2((m,C),(m_w,Cw)) = \| m - m_w \|_2^2 + \mathrm{Tr}(C + C_w - 2(CC_w)^{\frac{1}{2}}) \tag{7.4}$$

where:

$d^2((m,C),(m_w,Cw))$	= Squared distance between synthetic and real images
m	= Mean vector real
C	= Covariance matrix real
m_w	= Mean vector synthetic
C_w	= Covariance matrix synthetic

A mean feature vector (m or m_w) is computed by averaging the feature vectors extracted from all images processed by an Inception model up to a specified layer. Typically, activations from the "pool3" layer, one of the last pooling layers of the Inception network with a dimensionality of 2048, are used as feature representations for individual images. These feature vectors effectively capture high-level semantic information about the images, allowing for a meaningful comparison between the distributions of real and generated images.

In this investigation, the TorchMetrics package developed by Lightning AI [106] is used because it also supports multi-Graphics Processing Unit (GPU) computation, which is important since most of the computations are performed on the High Performance Computing (HPC) cluster at TU Dresden. It uses the Inception-v3 network [107]. Since the Inception network was trained on color images with an input dimension of 299 x 299 pixels, grayscale images are converted to RGB and resized to fit these dimensions within the framework.

The original implementation of the FID score for PyTorch by Seitzer [108] reports that at least 2048 images are needed to achieve a reasonable score, and the literature uses 10 k to 50 k images [109, 110]. While the FID score provides a quantitative measure of similarity between generated and real images, manual inspection allows for a qualitative assessment of visual quality. It can reveal subtle imperfections or artifacts that may not be captured by quantitative metrics alone.

7.2.4 Datasets

This section presents the three datasets used to train the GAN. These datasets are used to validate the reimplementation and to explore the behavior of CUT with different types of inputs, such as labels only, noise, multiclass labels, and a simulated dataset. When referring to a "label" or "mask" image, it contains pixel-wise classification annotations. In this type of image, each pixel value represents a specific class (e.g., 0 for background, 255 for foreground).

Horse2zebra Dataset

To ensure the accuracy of the CUT model and to streamline the debugging process, it is essential to first verify the reimplementation. The horse2zebra dataset from CycleGAN's publication (Figure 7.4) serves this purpose well, as it allows comparison with the original results presented in [5]. In this study, images of horses are transformed into zebras. The dataset contains 1187 horse images and 1474 zebra images for training, and 120 horse images and 140 zebra images for testing.

Figure 7.4: Examples from the horse2zebra dataset. **(a)**: Horses. **(b)**: Zebras. The images have dimensions of 256 x 256 pixels each.

Synthetic Concrete Pores Dataset (Synth-CP Dataset)

The Synthetic Concrete Pores (Synth-CP) dataset is derived from a subset of the Concrete Pores Segmentation dataset [46] presented in Section 5.2.2, with dimensions of 1536 x 1536 x 1024 voxels (depth (d) x height (h) x width (w)) and a voxel size of 9.3 µm. This dataset includes images extracted from a CT reconstruction (Figure 7.5c, target domain), binary masks representing the pores (Figure 7.5a, input domain 1), and binary masks with added noise (normal distribution with $\mu = 124$ and $\sigma = 25$; Figure 7.5b, input domain 2) to investigate the effect of a random input image background on the generated images. The binary pores were extracted from the CT reconstruction using a simple thresholding method.

This dataset is created because the domain translation is rather simple, since only the background needs to be adjusted. The presence of paired images with their masks also allows direct comparison of the generated images with their expected real-world counterparts. The dataset can be used both as a 2D and 3D version and is therefore also suitable for verifying a 3D version of CUT.

Synthetic Strain-Hardening Cement-Based Composite (Synth-SHCC) Dataset

The third dataset contains 10 simulated CT reconstructions of water-reduced SHCC, each with dimensions of 512 x 512 x 512 voxels (totaling 5120 images) and a targeted voxel size of 4 µm. The original CT reconstruction, presented in Section 5.2.3, focused solely on the segmentation of fibers. Here, a fully Synthetic Strain-Hardening Cement-Based Composite (Synth-SHCC) dataset is generated based on attributes from the real CT reconstruction, such as noise level, intensity, and constituents (Figure 7.6). This process aims to minimize the initial domain gap between synthetic and real data. Unlike the previous dataset, this one lacks paired images, making it a semi-supervised use-case without ground truth images, similar to the approach presented in [63], albeit in a different domain.

The dataset is created by adding the constituents in a predetermined order: clinker particles are added first, followed by quartz sand particles, metal contaminants, fibers, and finally pores. This sequential order ensures that the pores are in the foreground, allowing the fibers to traverse the other particles

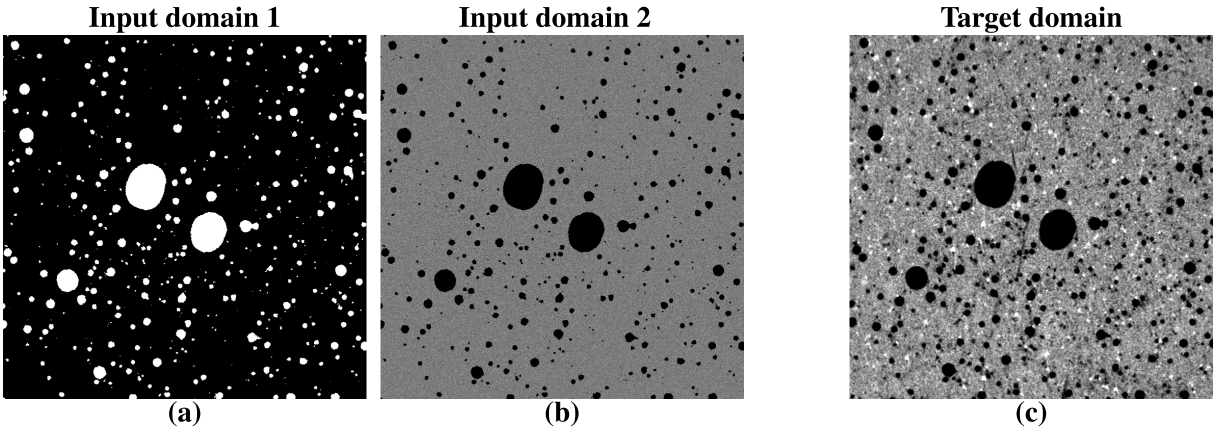


Figure 7.5: Examples of the Synth-CP dataset. **(a)**: Binary masks as input domain. **(b)**: Masks with noise as input domain. **(c)**: Target domain.

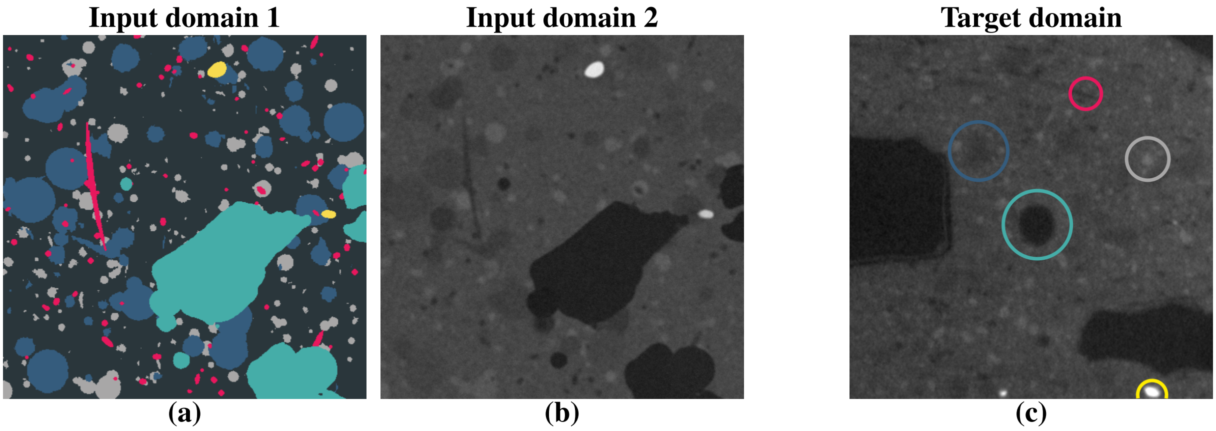


Figure 7.6: Examples from the Synth-SHCC dataset. **(a)**: Simulated masks as input domain. Visualized as a colored version for easier differentiation between the components: teal: pores; blue: quartz sand; gray: clinker particles; pink: polyethylene fibers. **(b)**: Simulated CT data as input domain. **(c)**: Real slice of a CT reconstruction (target data) with: teal: pore; blue: quartz sand; gray: clinker particle; pink: polyethylene fiber; yellow: metal contaminant. Due to the water-reduced nature of this concrete, the pores vary in shape and are not always spherical.

without significant disruption. This approach largely preserves the three-dimensional connections of the fibers and pores, enhancing the realism of the simulated dataset.
The volumes are created as follows:

1. Extracting pores from a real CT scan
   - Air is less dense than other components of SHCC and can be extracted using adaptive thresholding.
   - Store each instance of a pore from the generated binary volume.
2. Extract metal contaminant from a real CT scan
   - A metal contaminant is much denser than its surroundings, so it can be extracted using thresholding.
   - Store each instance of a contaminant from the generated binary volume.
3. Create an empty volume with user-defined dimensions (512^3 voxels).
   - This volume serves as a canvas for the subsequent generation of particles and objects.
4. Add clinker particles
   - Visually, clinker particles are generally similar in shape to pores, but smaller.
   - Load the pores, resize them to expected sizes (randomly selected within predefined ranges

based on a given voxel size), and apply random rotations.
    - Randomly distribute the particles throughout the volume.
5. Add quartz sand particles similar to clinker particles
6. Add metal contaminants similar to clinker particles, but with instances from step 2
7. Add fibers
    - Randomly choose a starting point for each fiber.
    - Define a spherical coordinate system around this point.
    - Randomly select an initial fiber development angle and a random initial curvature angle.
    - As the line length increases, the probability that the curvature angle will change at each step increases to introduce random curvature variations.
    - Randomly stop after a certain minimum length is reached.
    - Increase fiber thickness based on voxel size (requires knowledge of real fiber thickness).
8. Add pores by randomly resizing and rotating them
9. Add noise to the volume followed by blurring to simulate the noise in CT reconstructions
10. Save images (Figure 7.6b) and labels (Figure 7.6a)

7.3 Experimental Design

The core concept to perform an end-to-end semi-supervised or unsupervised training using CUT and the GCN consists of 3 main steps: 1. Label generation; 2. Training and inference of a CUT model using unpAIred; 3. Training and inference of a GCN model using AiSeg as described in Chapter 4.

In step 1, labels can be simulated or manually created as described in Section 7.2.4, and images of the target domain must be available. In step 2, a CUT model is trained on the data from step 1. The model is then used to generate synthetic training data. In step 3, the generated data can be manually curated and a GCN model is trained on the synthetic training data. The trained CNN is then applied to the CT reconstructions. When human interaction is required, such as manual curation of the generated data, the procedure is referred to as semi-supervised. If no human interaction is required other than the initial data preparation for the GAN, the procedure is called unsupervised.

The experiments were conducted on the HPC system at TU Dresden with the following resources for each training: 12 Cores @2.3 GHz, 24 GB RAM, 8 x NVIDIA A100-SXM4 (40 GB Video Random Access Memory (VRAM) each).

Training and Inference of CUT Models using unpAIred

Regarding CUT, all hyperparameters were adopted from the original implementation [5] unless otherwise stated:

In the Synth-CP dataset, the input/output channels were configured for grayscale images with an input dimension of 1024 x 1024 pixels. Two models, trained on binary and noise images, underwent 100 epochs of training with active online augmentation using a linear learning rate scheduling method. This scheduling helps to stabilize and optimize the training process, allowing faster convergence and better performance by avoiding overshooting or local minima traps [111]. The linear scheduler consists of a warm-up phase, a constant phase, and a decay phase. During the warm-up phase, the learning rate increases linearly with each epoch until it reaches the user-defined rate. The constant period maintains the user-defined learning rate, while the decay period linearly decreases the learning rate until the final epoch. Unlike the original CUT model, which had a constant learning rate for 200 epochs before decaying to nearly 0 by epoch 400, the models were trained for 100 epochs with a constant period of 75 epochs and a decay period of 25 epochs, assuming that the GAN would converge faster due to the higher input image resolution (4 x). Online augmentation techniques were applied to both the input and target domains with a global strength of 100 %, including random flipping (75 %) and resizing (33 %; scaling randomly ranges from 1.0 to 1.2 to avoid mirror effects).

During inference, training data (images and masks) for two synthetic concrete pores segmentation datasets were generated. The first dataset contained binary images of input domain 1 as the ground truth and images generated using input domain 2 (Section 7.2.4), which served as input to the GCN. The second dataset was generated by applying the CUT generator to the full Concrete Pores Segmentation dataset [46] (introduced in Section 5.2.2). All ground truth masks from the dataset were used to generate realistic images for the target domain of the Synth-CP dataset (Figure 7.5c).

Regarding the Synth-SHCC dataset, the input dimension was set to 512 x 512 pixels to match the dimension used to simulate the input data for the GAN. Two models, using multiclass labels and simulated data, were trained for 200 epochs instead of 100 because the target domain was more complex. Both trainings ran with a constant learning rate period of 125 epochs followed by a decay period of 75 epochs. The same online augmentation settings were used as for the Concrete Pores Dataset.

During inference, training data for a SHCC pores segmentation dataset was generated. It comprised pore labels from the Synth-SHCC dataset as ground truth (binary segmentation) and images generated using CUT on the input domain of the Synth-SHCC dataset (Figure 7.6b).

Finally, the three generated datasets were verified by comparing the output images of CUT to their ground truth masks. Images that did not visually match the shapes in the ground truth were removed.

Training and Inference of GCN Models using AiSeg

The hyperparameters of the GCN on the three datasets were almost identical. They differed only in the input size (SHCC: 512 x 512; Concrete Pores (CP): 1024 x 1024), the batch size (SHCC: 48; CP: 12), and the total number of epochs (SHCC: 250; CP: 100). This difference was due to the fact that the synthetic pore segmentation datasets contained many more images than the synthetic SHCC pore segmentation dataset, and therefore the model was expected to converge earlier. Due to the different total epochs, the learning rate scheduling and fine-tuning epoch differed (SHCC: warm-up: 3; constant: 147; decay: 100; fine-tuning: 200. CP: warm-up: 3; constant: 67; decay: 30; fine-tuning: 85). The fine-tuning epoch disables online augmentation for the rest of a training. All trainings were started using a learning rate of 0.001, the AdamW optimizer, which is a fixed version of Adam [112], automatic weighting to prevent class-imbalance during optimization, and Group Normalization [113] as normalization layer.

While training the GCN on the synthetic datasets, online augmentation was performed as described in Section 4.4 using the settings presented in Table 7.1. Online augmentation was disabled when the fine tuning epoch was reached. Settings that are not shown have been disabled. The geometric operations were set so that the pores differed in shape, and the radiometric operations were set so that the general appearance contained more or less noise and was brighter or darker.

Parameter	Value	Parameter	Value	Parameter	Value
Policy	geo first	Global strength	85 %	Max augmentations	5
Max rotation angle	± 15°	Resize scale	1.0 to 1.2	Vertical flip allowed	Yes
Elastic distortion	33 %	Flip	75 %	Grid distortion	50 %
Rotate by angle	33 %	Resize	33 %	Brightness	50 %
Contrast	50 %	Gaussian filter	50 %	Gaussian noise	50 %

Table 7.1: Online augmentation parameters for the GCN on the via CUT generated training data.

7.4 Results and Discussion

7.4.1 Training and Inference of CUT

horse2zebra

The original implementation used the pytorch-fid package [108] to compute the final FID, which is different from that used in unpAIred. They got a score of 45.5. However, it is not clear whether the score is based on the test data, the training data, or both. Therefore, different combinations were tested using the checkpoint of epoch 349, which achieved a training FID score of 25.8 (TorchMetrics). To compute comparable results, the FID was computed again using the pytorch-fid package. Since the horse2zebra dataset contains fewer images than needed to compute a reasonable FID score (1307 (training + test) horse images in total), the FID score is not sufficient to make a general statement about the similarity between generated zebras and real images of zebras. However, it allowed for a comparison with the original implementation. The results are shown in Table 7.2.

FID train	FID test	FID train and test	FID (train + test) horses on test zebras
26.8	74.3	27.6	47.3

Table 7.2: FID computed using different subsets of the dataset. FID train: calculated using generated samples form the training horses against the training zebras. FID test: calculated using generated samples form the test horses against the test zebras. FID train and test: calculated using generated samples form the training and test horses against the training and test zebras (full dataset). FID (train + test) horses on test zebras: calculated using generated samples form the training and test horses against test zebras.

Table 7.2 shows very different FID scores. Assuming that the original implementation used only the test data, the reimplementation could not achieve the reported FID score (74.3 vs. 45.5). However, when the FID was computed using all the horse images and compared to the test data only, the score got closer (47.3 vs. 45.5), and when all the images were used, the FID dropped to 27.6, indicating strong overfitting on the training data. In the case of creating a dataset for semantic segmentation, this is not problematic, since the focus is on generating realistic images that mimic the target domain and are paired with the input images. In the case of image synthesis, however, this is not a desired behavior.

Differences between the reimplementation and the original version can be explained by the different initial random seeds as explained in Section 5.4.1, different PyTorch versions (this: 2.1; theirs: PyTorch 1.1), and different hardware settings (this: 8 x A100; theirs: 1 x GTX 1080Ti). Training on the HPC took only 2 hours and 49 minutes for 400 epochs (without FID computation).

Figure 7.7 shows the training loss and FID on the training data. The loss graph shows the effect of the fine-tuning epoch, which was set to 200 in the original implementation. After this epoch, the learning rate decreased and online augmentation was disabled. The variation of the FID score on the training data became smaller towards the end of training as the learning rate decreased. It reached a local minimum at epoch 349 and then increased until epoch 400. This suggests that a slight overfitting occurred on a subset of the data.

As described in Section 7.2.3, the FID alone is not sufficient, and human judgment was also required. Figure 7.8 shows examples. It can be observed that the model generally performed well when horses were in front of a green background (Figure 7.8b). In places where the training dataset did not contain many images, such as in front of a beach, the sky, or rocks, it performed worse (Figure 7.8d). It seems that the network is sensitive to the background and has not learned that the objective is to alter only the horses. Furthermore, this supports the hypothesis of overfitting on images with a specific image composition (horses in front of a green background), which can also explain the increasing FID score towards the end of training.

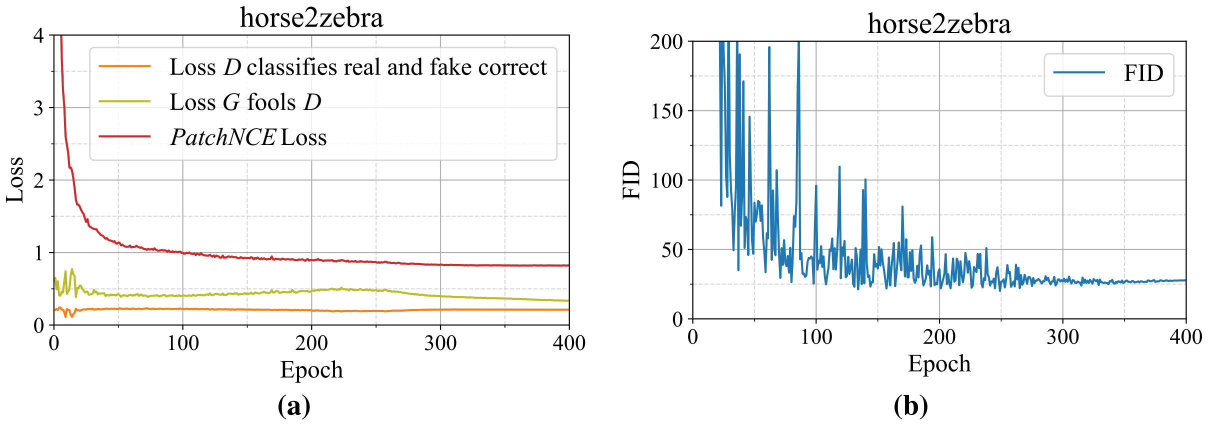


Figure 7.7: Training results of CUT (epoch 349) on the horse2zebra dataset. (**a**): Losses of CUT. (**b**): FID score.

In general, the results were visually very similar to the original implementation[1] and verified that the implementation produced meaningful results.

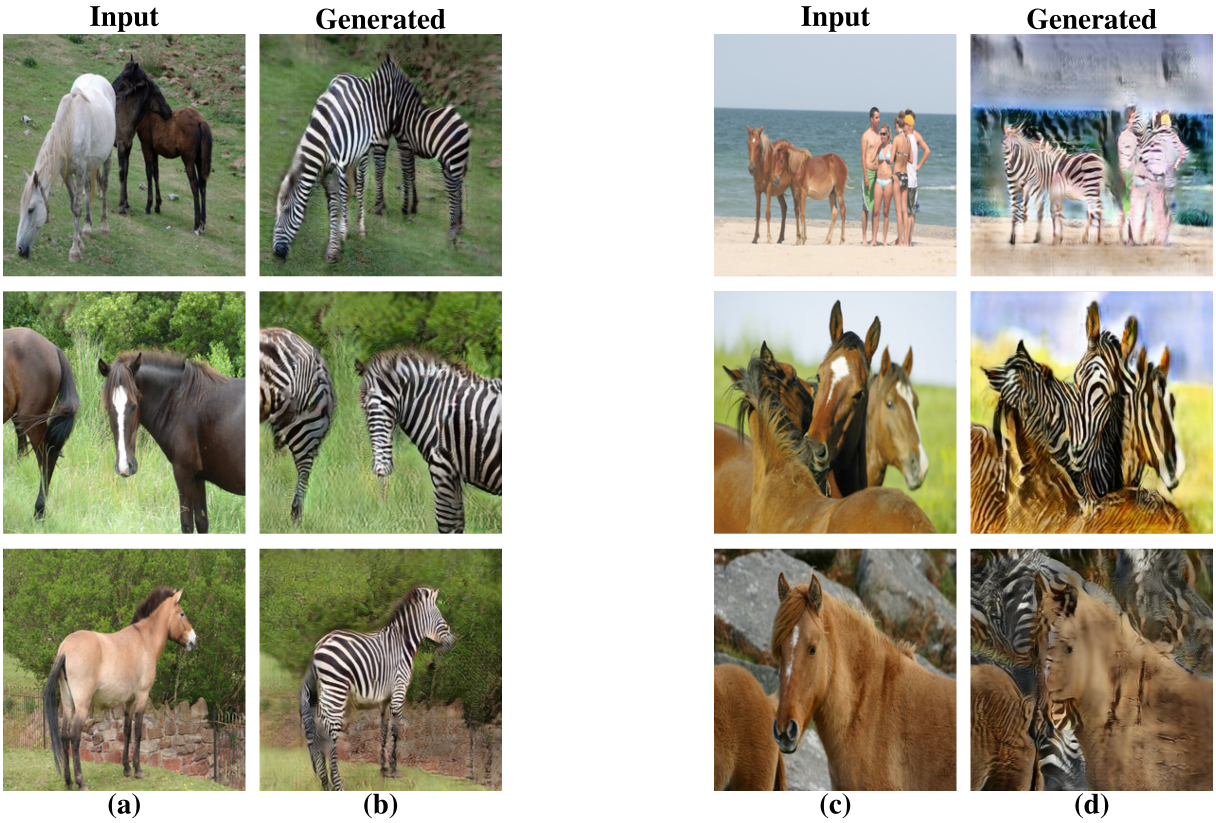


Figure 7.8: Example generations from the horse2zebra dataset (test images, epoch 349). (**a**) and (**b**): Input and generated images of successful translations. (**c**) and (**d**): Input and generated images of failed translations.

Synth-CP Dataset

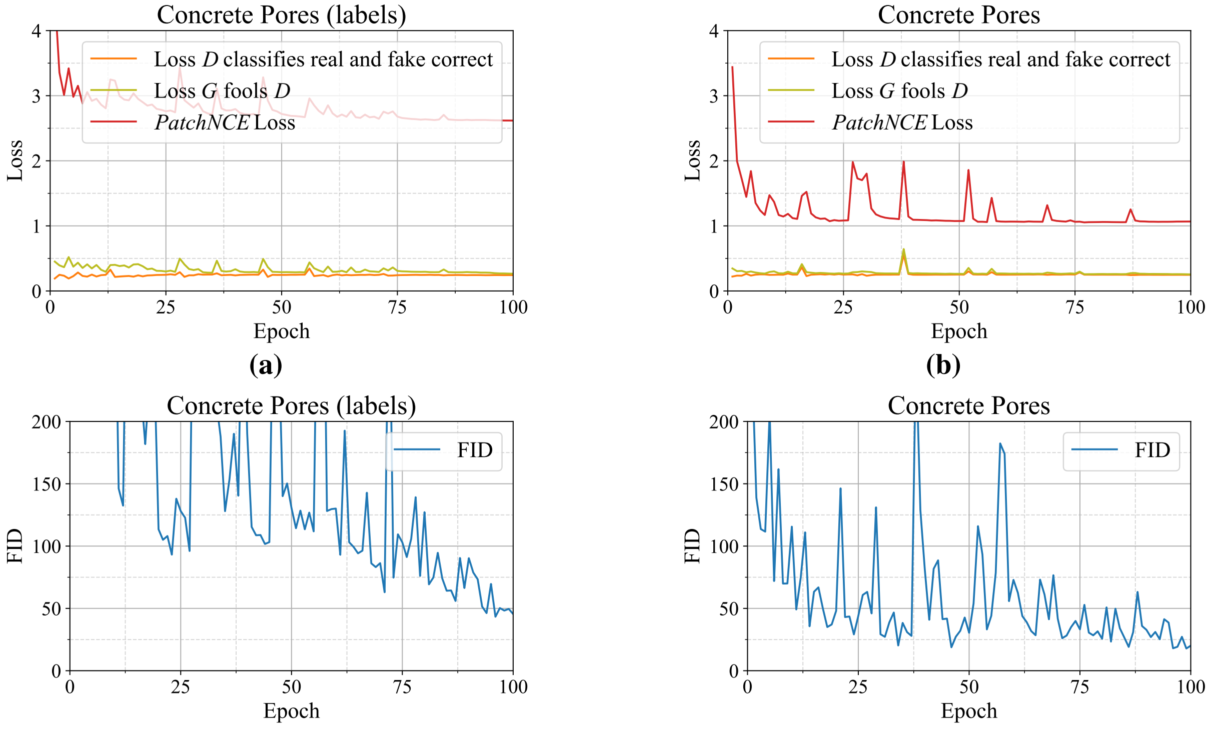
To verify the assumption that CUT hallucinated in regions where the input data contained homogeneous areas (Section 7.1), and to investigate whether binary images are sufficient, two trainings

[1] https://github.com/taesungp/contrastive-unpaired-translation

were performed: First, a training using binary labels, and second, inverted binary labels containing noise.

Training on the HPC for one model took 4 hours and 36 minutes for 100 epochs (without FID computation).

From the loss plots in Figure 7.9a and b, the large discrepancy between the *PatchNCE* losses is noticeable. This suggests that shapes from the input images were not included in the generated data when using only label images as input. Furthermore, the FID results in Figure 7.9c and d suggest that generating with binary labels only had a direct, negative impact on image quality. From these graphs alone, it can be concluded that random noise significantly improved the performance of CUT.



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

Figure 7.9: Training results of CUT on the Synth-CP dataset. **(a)** and **(b)**: Losses of CUT using labels as input (a) and noisy images (b). **(c)** and **(d)**: FID scores per epoch computed with 1024 images using labels as input (c) and noised images (d).

In Figure 7.10a, the input contained only labels, where 0 (black) was the background and 255 (white) represented pores. This is typical for a binary image segmentation dataset. However, the generated image in Figure 7.10b contains almost only hallucinated pores, which can be seen in the combined version (Figure 7.10c). Furthermore, in the top image of Figure 7.10b, artifacts are visible around the larger central pore. This is directly due to the fact that a large pore should be generated there. Therefore, this version was not usable for the generation of a semantic segmentation dataset.

In Figure 7.10d noise has been added to the binary labels and the pores were colored black. These small changes significantly improved the generated result (Figure 7.10e) not only in terms of the absence of artifacts, but also in terms of the positioning of the pores (Figure 7.10f).

The test shows that noise not only stabilized and improved training, but also led to better visual results. Therefore, the addition of noise was essential in generating a synthetic training dataset for image segmentation.

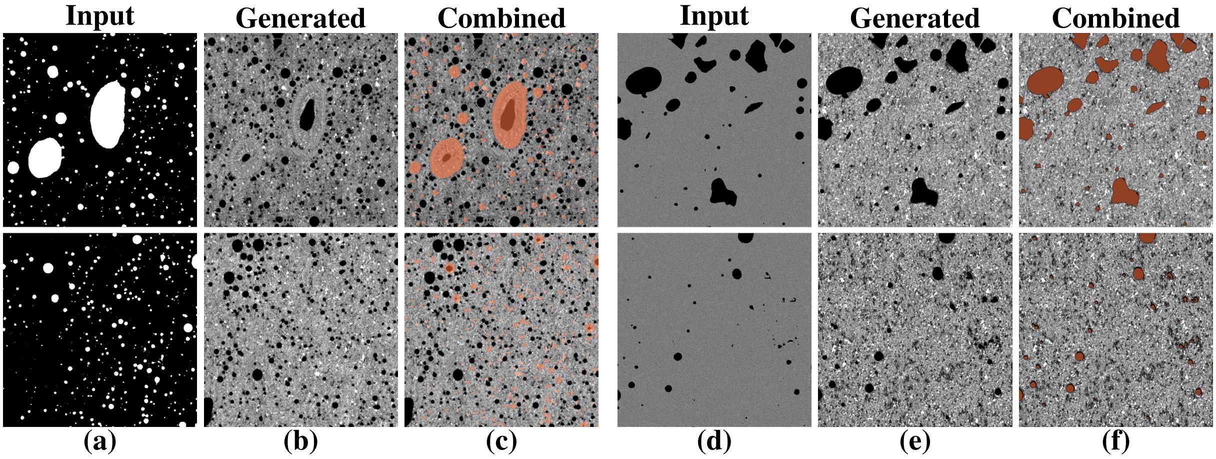


Figure 7.10: **(a)**, **(b)** and **(c)**: Input, generated and combined examples of generated concrete pores using binary labels as input. **(d)**, **(e)** and **(f)**: Input, generated and combined examples of generated concrete pores by adding noise to the input.

The FID on the generated data using noise with pytorch-fid was rather small with 29.9 and was computed using 3072 generated images and 3564 real images of the Synth-CP dataset (images were resized from 1024 x 1024 to 299 x 299). In Figure 7.11b and c, a generated image and its real-world target are shown side by side. The generated image had a mean gray value of 154.4 with a standard deviation of 75.8. The target image had a mean gray value of 167.3 with a standard deviation of 73.0.

The generated image had slightly higher noise levels compared to the target image, appeared darker, and lacked fibers (dark lines in Figure 7.11c). The darker appearance of the generation can be explained by the prevalence of darker images in the target domain, especially those in the center of the CT reconstruction. The center of the CT reconstruction may be darker than the outer regions due to cupping, as discussed in Section 3.3. The increased noise may be due to input noise during the training process. In addition, based on the FID score, it appears that the training may have been terminated prematurely. However, due to the learning rate scheduling, the training cannot be continued from epoch 100, but from epoch 75. Nevertheless, since the generations were initially satisfactory, this is a task for future studies.

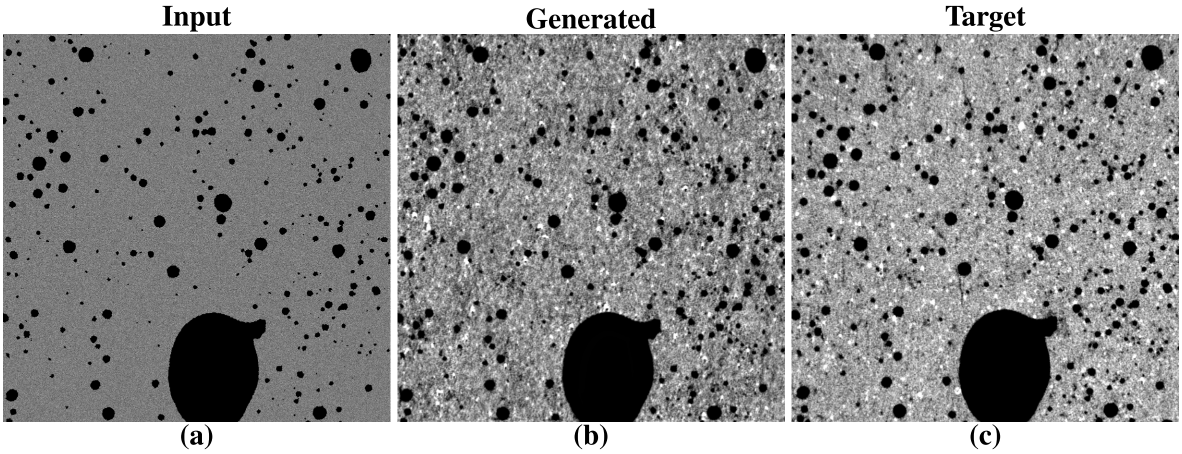


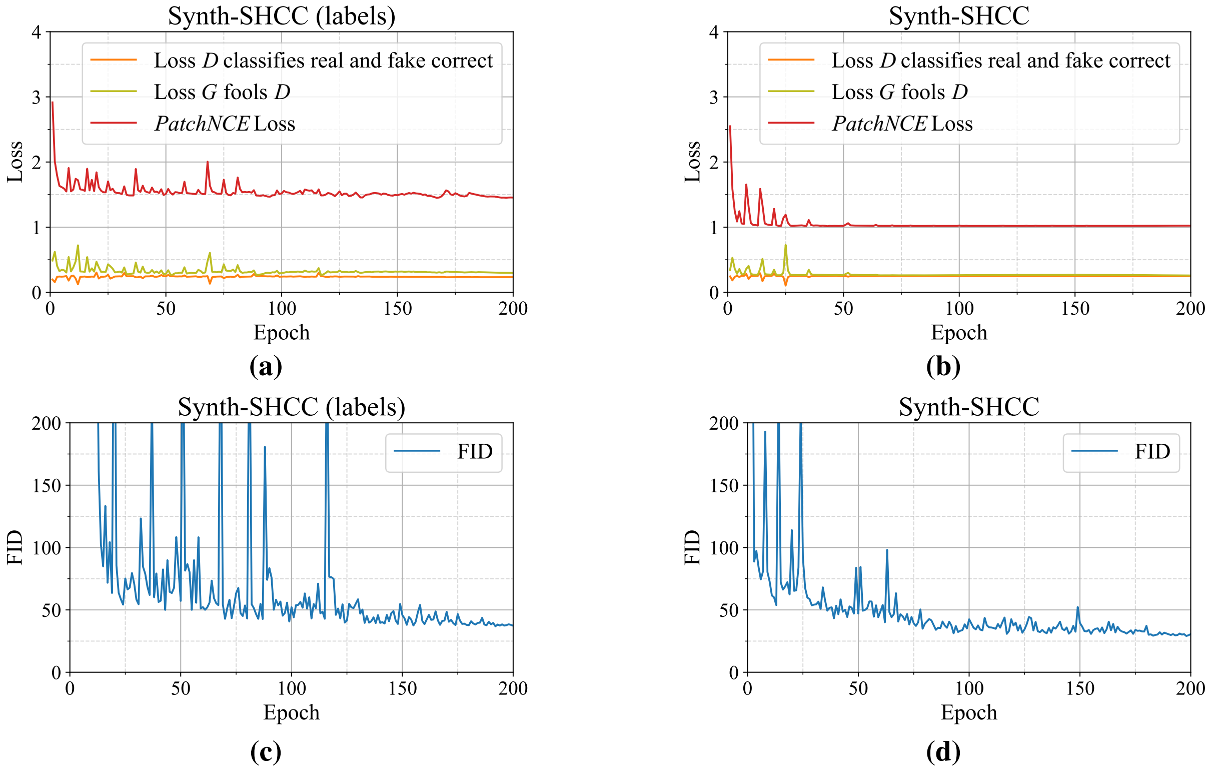
Figure 7.11: Input **(a)**, generation**(b)** and ground truth **(c)** of a sample from the Synth-CP dataset.

Due to the shape preserving output of the GAN, the model trained with noise was used to generate training data for a semantic segmentation task. In addition, to overcome the brightness differences and noise level, blurring and brightness changes were added to the online augmentation to train the GCN.

Synth-SHCC Dataset

In the Synth-SHCC dataset, the labels are more complex, such as in the case of the Cityscapes dataset performed by Park et al. [5]. To ensure and reproduce that the errors that occurred when using only labels without noise as input were not simply due to the binary nature of the labels, the test was also performed on the multi-class Synth-SHCC dataset: First, a training with only labels, and second, a training with simulated CT images.

The loss curves (Figure 7.12a and b) show a similar pattern as in section 7.4.1. The use of noise and more realistically shaded constituents helped to stabilize the training and reduce the training FID (Figure 7.12c and d).



(a)

(b)

(c)

(d)

Figure 7.12: Training results of CUT on the Synth-SHCC dataset. **(a)** and **(b)**: Losses of CUT. **(c)** and **(d)**: FID scores calculated during the training with 1024 images.

Figure 7.13 reinforces the assumption that labels alone were insufficient as input. However, the images generated from the simulated input do not show the same degree of alignment as previously observed. Pores often appear larger (red arrows in Figure 7.13c), and the GAN hallucinated pores, a trend observed in many generated images.

The FID of the simulated data using 3972 real images of a CT reconstruction of SHCC and 5120 generated images (epoch 117) computed with the pytorch-fid library was 82.5. This is the highest value of all datasets and implies a large domain gap. However, the mean gray value of the generated images was 69.5 (σ of 14.5), which is close to the target with a mean gray value of 67.6 (σ of 16.1). Since the generations appear close to the target images, the gap may be caused by misshaped clinker and quartz sand particles, overlapping constituents (e.g., in the center of the bottom images in Figures 7.13c and d, where a clinker particle overlaps a quartz sand particle), or different pore dimensions.

The described hallucinations and shape inconsistencies are likely to negatively affect the performance of a CNN trained on synthetic semantic segmentation data for SHCC. Therefore, a dataset has to be

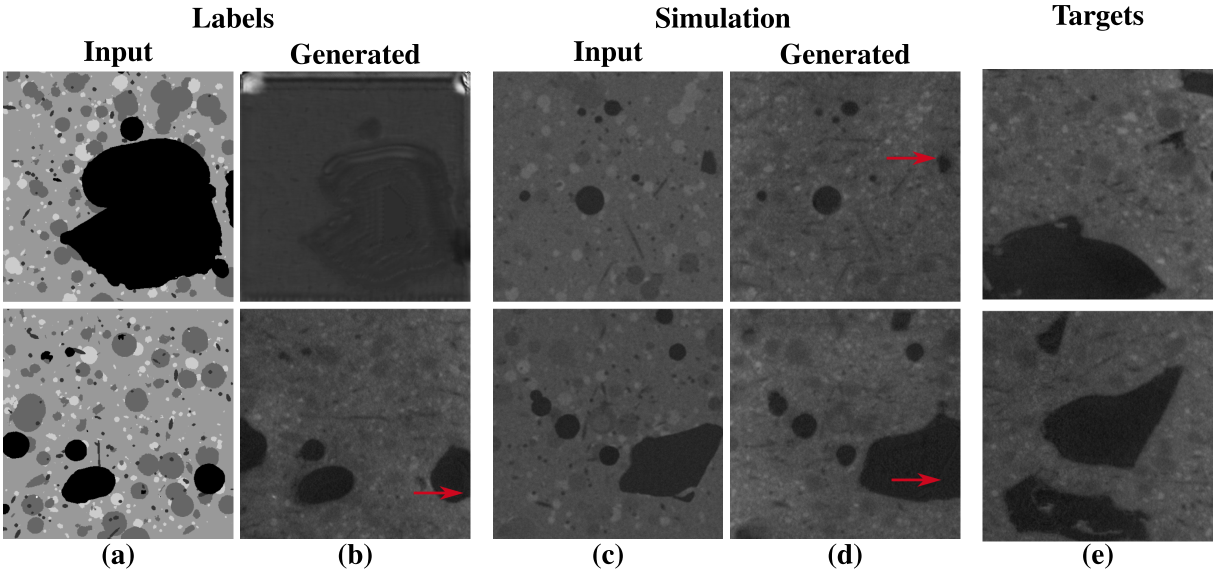


Figure 7.13: Examples of generated SHCC with pores using binary labels ((**a**) and (**b**)), simulated data ((**c**) and (**d**)), and real images (**e**). Smaller errors are marked by red arrows.

carefully supervised to remove insufficient generations.

7.4.2 End-to-End Training for Semantic Segmentation

After training GANs for the creation of training data for synthetic pores segmentation datasets, validation and test data was added in order to create complete datasets for AiSeg. Afterwards GCN models were trained.

Segmentation of Pores using the Synth-CP dataset

Two synthetic datasets were generated for the segmentation of pores in the first reconstruction containing concrete: a large dataset and a small dataset.

In the case of the large dataset, the training data consisted only of labels from the Concrete Pores Segmentation dataset [46]. These labels were preprocessed in the same way as input domain 2 of the Synth-CP dataset, resulting in training data of 8005 samples (consisting of images and corresponding masks). For validation, a subset of the original CT reconstruction was used, consisting of 1600 samples, each with a size of 1024 x 1024 pixels. In this context, the validation images also served as test data, since no hyperparameter tuning was performed.

As for the small dataset, it contained the same test and validation samples, but only 3072 training samples. These samples were generated exclusively from the images of input domain 2 of the Synth-CP dataset. With an input dimension of 1024 x 1024 pixels, each original image (1536 images with dimensions of 1536 x 1024 pixels each) was sliced into two overlapping images.

The training data of both datasets remained unchanged and unverified, simulating unsupervised training on synthetic data. The only supervised action was to extract the labels from the Concrete Pores Segmentation dataset. The training duration for the large dataset on the HPC was 33 hours and 30 minutes, and for the small dataset, it was 15 hours and 46 minutes.

Analyzing the loss graph of the smaller, domain-specific dataset (Figure 7.14a), no overfitting is evident. At epoch 45, the minimum validation loss of 0.052 was reached, suggesting that early stopping could have potentially reduced the training duration. However, due to the fixed learning rate scheduling, the total training duration was fixed. By epoch 100, the validation loss increased slightly to 0.059. Regarding the validation accuracy (percentage of correctly classified pixels) (Figure 7.14b), epoch 45 also yielded the highest value of 98.78 % (epoch 100: 97.96 %). Consequently, epoch 45 was considered the best checkpoint and was used to calculate the DICE coefficient that was introduced in Section 5.3.3. The calculation was performed on the test data (corresponding to the validation data) after training. The model achieved a DICE coefficient of 93.83 %. Given the alternating characteristics of the validation accuracy (Figure 7.14b), it appears that the learning rate may have been slightly too high for this training and could be reduced in future iterations.

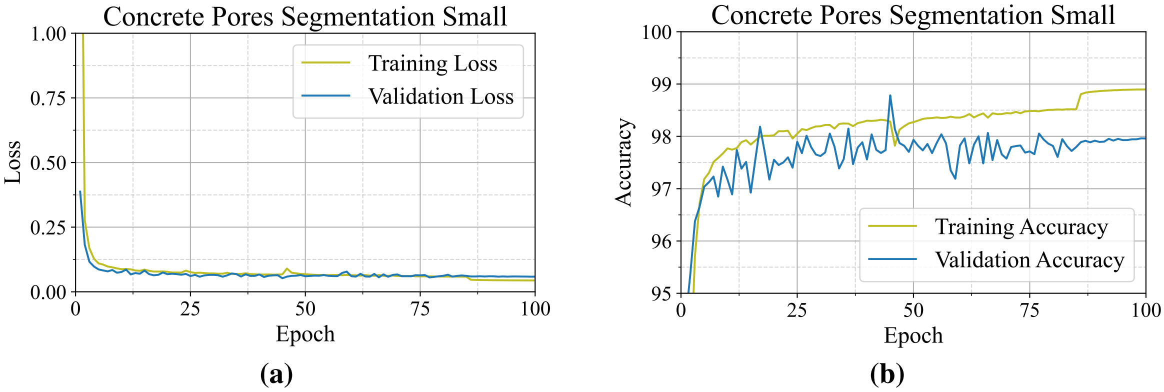


Figure 7.14: Training and validation loss **(a)** and accuracy **(b)** of the GCN (ResNeXt 50) on the small synthetic concrete pores segmentation dataset.

The loss graph of the larger dataset (Figure 7.15a) shows a slight overfitting on the training data, which was unexpected considering that the training data was expected to have a higher variance than the smaller dataset. A peak occurred at epoch 48, likely due to the use of the AdamW optimizer.

AdamW adaptively adjusts the step size based on past gradients, which can occasionally result in overly aggressive updates. If the step size becomes too large, the optimizer may overshoot the optimal solution, leading to an increase in loss. Another contributing factor is the use of an exponential moving average in the denominator term. As gradients decrease, the denominator decreases as well. Consequently, with already small gradients, the denominator can amplify the effect, potentially causing large updates and pushing the optimization process further away from the optimal solution, resulting in a notable increase in loss [114, 115].

After this peak, the validation loss did not decay back to the previous minimum, which occurred at epoch 47 with a value of 0.078 (accuracy: 97.29 %). Therefore, epoch 47 was chosen for a first test of the CNN. It achieved a DICE of 87.50 %, which is significantly lower than the DICE of the small dataset. Since the accuracy increased steadily until epoch 100, epoch 100 was also tested to confirm that epoch 47 performed best. At epoch 100, the model achieved a DICE score of 89.06 %. Although higher, this score was still worse than the scores obtained with the small dataset. However, it showed that using the lowest loss as an indicator for the best model alone is not sufficient.

The lower scores can be explained by the more diverse data. The model may have been more general and applicable to other CT reconstructions of this type of concrete, but in this scenario the CNN was trained to segment pores of a specific reconstruction with a fixed voxel size, and therefore the model trained on the small data was superior.

In both trainings, the effect of fine-tuning after epoch 85 and learning rate decay after epoch 70 can be observed. After epoch 85 (fine-tuning), the training loss decreases slightly and the training accuracy increases abruptly. The deactivation of the online augmentation therefore led directly to a stronger

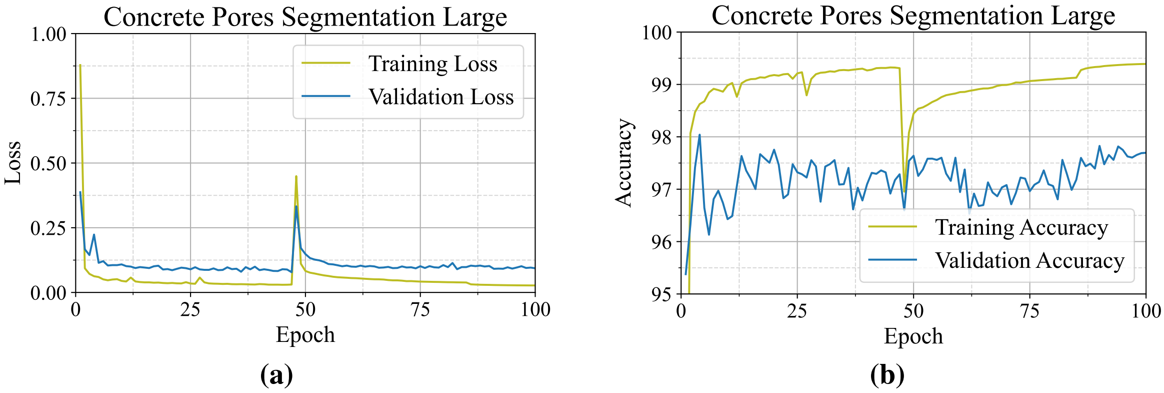


Figure 7.15: Training and validation loss **(a)** and accuracy **(b)** of the GCN (ResNeXt 50) on the large synthetic concrete pores segmentation dataset.

overfitting, while the learning rate decay (after epoch 70) contributed positively by slightly improving the accuracies and of course by reducing their alternating characteristics due to smaller updates by the optimizer. Therefore, in future training, online augmentation should remain active and the learning rate decay period can be increased.

In Figure 7.16, the CT reconstruction, which was the target of the pore segmentation using the Synth-CP dataset, has been segmented and visualized using the GCN trained on the small dataset. On the right, a horizontal slice of the volume is shown and the pores are highlighted. The red circles show 3 examples of the remaining errors. The errors occur only for very small pores.

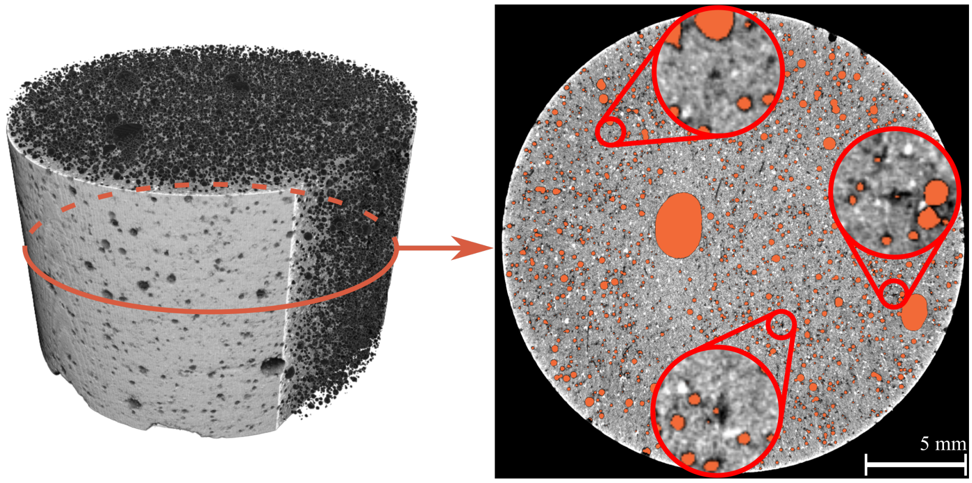


Figure 7.16: Segmentation of pores in a concrete sample using a GCN trained only on synthetic training data. Pores are colored black (left) and orange (right) for visualization. The GCN failed to segment very small pores (red circles).

Segmentation of Pores using the Synth-SHCC Dataset

The generated SHCC training data exhibited numerous shape inconsistencies, as elaborated in Section 7.4.1. Consequently, out of 5120 generated images, only 1558 samples proved suitable for training.

The test data was the same as the validation data, for the same reasons as in Section 7.4.2, and consisted of 60 images from the original fiber segmentation dataset, where the pores were manually segmented. Due to the batch size of 48 and 8 GPUs, 60 images were not sufficient for the validation. Therefore, the validation data was augmented to contain 384 images using geometric techniques such as rotating, flipping, grid distortion, and radiometric augmentations such as blurring, noise addition, and contrast manipulation.

Despite the supervised curation, the generated samples still contained errors, as illustrated in Figure 7.17, and did not contain much variety (Figure 7.18a): The generated data contained many images that were very similar, which was due to the fact that the input data was extracted from 3D volumes, where no abrupt changes from one depth slice to another happened. In fact, this was also the case for the Synth-CP dataset, but due to the voxel size of 4.0 µm of the Synth-SHCC dataset, a pore was more than twice the size of a pore from the Synth-CP dataset (voxel size of 9.3 µm) and thus distributed over more slices. If the GAN performed well on one slice of a simulated volume, it was likely that the next slices were equally well generated. When removing all but one instance of a stack (Figure 7.18b), the dataset would be reduced to only 83 different training images. However, all images were used because it was assumed that the variance of subsequent slices can be sufficiently increased by online augmentation and because small differences from slice to slice were still apparent.

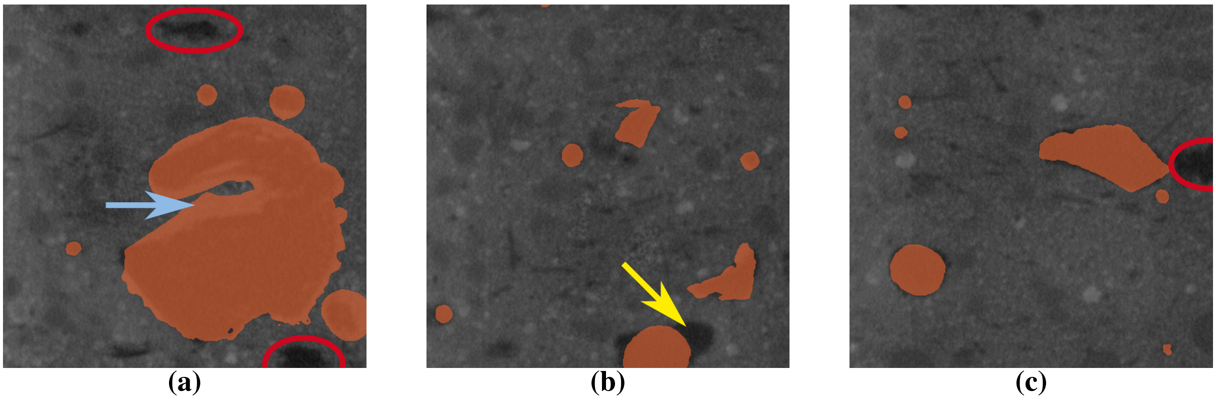


Figure 7.17: Combined inputs (images) and targets (orange) from the training data. Red: hallucinated pores; Light blue: Shape inconsistency, where the GAN generated a too small pore. Yellow: Shape inconsistency, where the GAN generated a too large pore.

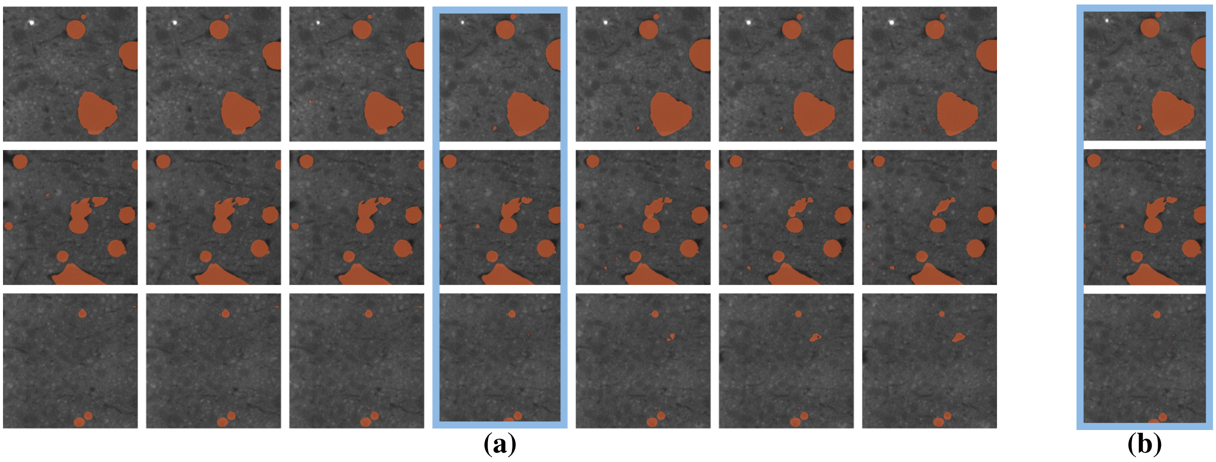
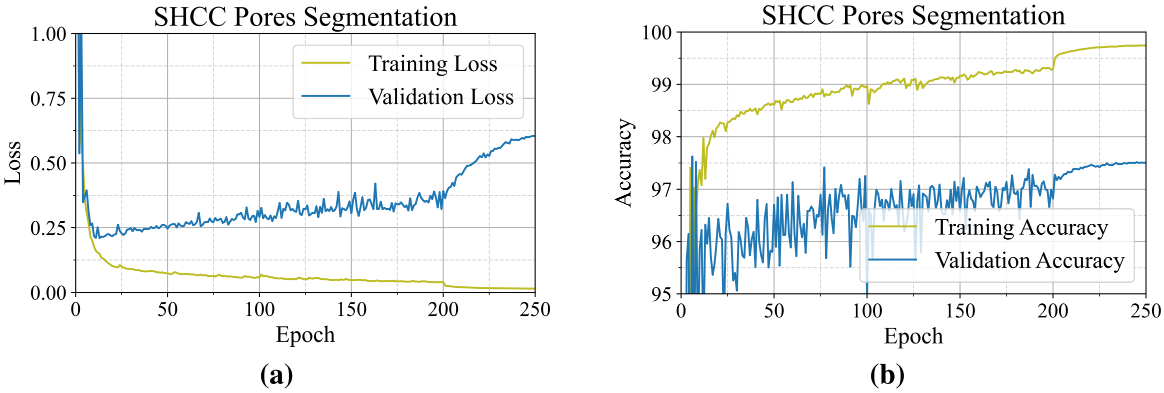


Figure 7.18: Successive slices of combined inputs (images) and targets (orange) from the training data **(a)** and scene reduction **(b)**.

The training duration for the dataset on the HPC was 21 hours and 5 minutes. From the loss graph (Figure 7.19a), a strong overfitting on the training data can be observed, which can be explained by the rather invariant training data.

At epoch 13, the minimum validation loss of 0.210 was reached with an accuracy of 96.32 %. The model trained to this epoch achieved a DICE score of 80.63 %. At the end of both graphs (Figure 7.19), it can be observed that deactivating online augmentation increased the validation accuracy and loss. This means that the model produced very little but large large errors, as explained in Section 5.3.3, the checkpoint trained to the last epoch could possibly achieve a better DICE score. Therefore, epoch 250 was also tested on the unaugmented data. This model improved the DICE score to 82.95 %.



(a) **(b)**

Figure 7.19: Training and validation loss **(a)** and accuracy **(b)** of the GCN (ResNeXt 50) on the synthetic SHCC pores segmentation dataset.

Although the DICE score was improved by using the final checkpoint, the score was still rather low when compared to the results obtained on the concrete pore segmentation dataset from Section 5.4.3. In that section, a 3D U-Net achieved a DICE score of 87.66 % without augmentation and on test data that contained much more variety. Of course, this was also due to the fact that it was a 3D CNN and therefore could handle the 3D context better, but the goal for the GCN was to achieve similar or better results on this specifc CT reconstruction, since it was trained specifically on this data.

In Figure 7.20, the CT reconstruction containing the entire SHCC volume has been segmented and visualized using the GCN trained on the synthetic SHCC pore segmentation dataset. A longitudinal slice is shown on the right side of the image and errors are highlighted. Although not perfect (Figure 7.20, red and yellow arrows), most of the pores were successfully segmented. However, some thicker fibers are also incorrectly segmented (Figure 7.20 yellow arrow).

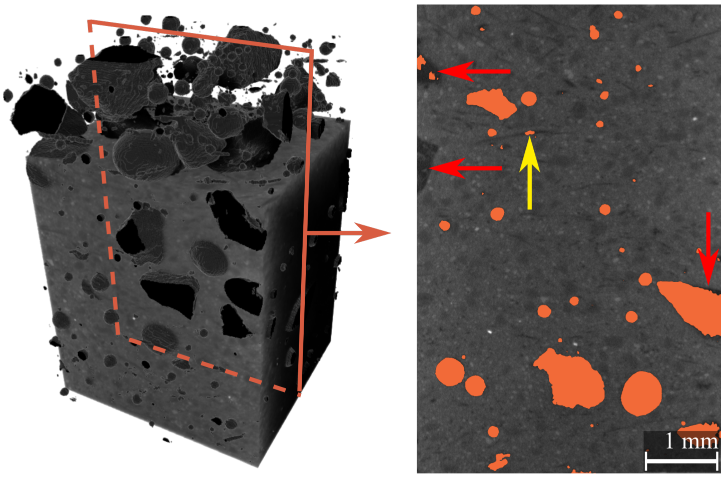


Figure 7.20: Segmentation of pores (black) in a sample of SHCC using a GCN trained only on synthetic training data. Pores are colored black (left) and orange (right) for visualization. The GCN failed to segment some pores (red arrows) and misclassified fibers as pores (yellow arrow).

7.5 Conclusion

In this chapter, the potential of using a shape-preserving GAN called CUT to generate synthetic training data for semantic segmentation of pores in CT reconstructions of concrete via end-to-end semi-supervised training was demonstrated. CUT was used to generate synthetic training data, which was then used to train GCN, a convolutional neural network tailored for image segmentation.

Two different CT reconstructions were used to perform unsupervised and semi-supervised segmentation. The first reconstruction contained concrete with a fairly homogeneous background. Pores could be easily segmented by thresholding, which allows to verify the result of the procedure. The second reconstruction contained a different type of concrete, called Strain-Hardening Cement-based Composite (SHCC). This reconstruction contained more distinguishable constituents and more complex pores due to the water-reduced concreting process.

To synthetically generate training data, labels representing pores were generated for each of the reconstructions. CUT generated realistic images that were almost indistinguishable from real CT reconstructions. It was observed that artifacts in the generated images were caused by overly uniform labels. Therefore, the input images needed to have some structural complexity to avoid hallucinations and inconsistencies in the shape of the objects of interest. "Structure" in the form of simple noise was used for the first reconstruction, and a more complex simulation aimed at mimicking SHCC was used for the second reconstruction.

It was shown that simple noise can be sufficient to generate realistic synthetic images from CT reconstructions that do not contain much information. The synthetic samples of the first reconstruction contained deceptively real pores and concrete, with the location of the input pores perfectly contained in the generated images.

The second, more complex reconstruction (SHCC sample) proved to be more challenging. Although the generated images were visually realistic, the shapes of the input pores were not fully preserved. In some cases, pores were hallucinated, too small, or too large.

Despite these challenges, the generation of training data proved successful, and the use of online augmentation complemented the training of GCN models on both synthetic datasets. The first dataset targeting the simpler reconstruction could be used in an unsupervised manner, eliminating the need for human interaction to train the GCN. Conversely, the data generated for the second reconstruction required supervised curation by removing samples where the targets (pores) didn't match the generated images. Fortunately, a supervised cleanup process took only 10 minutes because many subsequent images contained the same errors, significantly less time than manual data generation. However, even with supervision, the GCN struggled with accurate pore segmentation, missing some pores and misclassifying voxels in regions containing thicker fibers.

The final results of the segmentations were visually promising. The first CT reconstruction could be segmented almost perfectly, whereas the result on the SHCC reconstruction could be enhanced.

These findings underscore the need to refine the synthetic dataset generation process. Nevertheless, this chapter demonstrates that a combination of online augmentation and synthetic training data generation can be successfully used for voxel classification, resulting in end-to-end semi-supervised training.

7.5.1 Future Work

This chapter opens the door for further exploration of semi-supervised segmentation within CT scans, such as fiber segmentation in SHCC. Visual tracking of fibers in successive images of volumes from the generated Synth-SHCC dataset shows minimal fiber hallucination, and the fibers show continuous motion throughout the volume.

Another potential application is the segmentation of carbon rovings in carbon reinforced concrete. However, this task can be challenging due to subtle gray value differences between carbon and air, making it more likely for CUT to hallucinate. However, when scanning a carbon roving without concrete, the GAN only needs to generate the concrete around the roving, simplifying dataset generation by eliminating manual modeling of constituents.

A third possibility is unsupervised crack segmentation, where the combination of this approach with an Extended Finite Element Method (XFEM) for crack simulation [116] could automate crack generation in 2D. In a subsequent study by Krüger et al. [117], the authors highlight the significant reduction in computational complexity and conclude that this is particularly beneficial for 3D simulations.

Therefore, the logical next step is to extend CUT to 3D. 3D variants of CUT's predecessor, CycleGAN, have already been published [118–120]. They have demonstrated the feasibility of volume-to-volume translation. However, the 3D-CycleGAN requires a significant VRAM allocation, with 32 GB VRAM needed for input dimensions of only 64 x 64 x 64 voxels. Small input dimensions, especially in the context of 3D CT reconstructions, can result in the loss of important contextual information that is either truncated or not preserved. Furthermore, GANs are very sensitive to hyperparameter adjustments and CUT, in contrast to CycleGAN, seems to be much more sensitive to changes in the dimension of the convolutions. As initial tests have shown, a simple change from 2D convolutions to 3D convolutions leads to mode collapse and the so-called catastrophic forgetting [104].

Finally, a possible improvement for AiSeg emerged during the training of the GCN: a combination of automatic stop condition and learning rate decay: When a stop condition is reached, e.g. when certain metrics stop improving, an automatic learning rate decay takes place.

Chapter 8

Synthesis

The study presented conventional 2D and 3D augmentation methods in the context of training neural networks, and were extended by a deep learning based approach called Contrastive Unpaired Translation (CUT). They all contribute to one of the core problems of machine learning: data scarcity, and allow neural networks to be trained with little or no labeled data. It has been shown that the performance of a Convolutional Neural Network (CNN) for semantic segmentation can be effectively improved with respect to computed tomography and other domains. In this section, the chapters are shortly summarized and the main findings of their discussion are presented and placed into the context of this work. Afterwards, potential improvements based on overarching findings are presented. Finally, a conclusion of the work is presented, which completes the dissertation.

8.1 Research Summary

8.1.1 Augmentation Pipelines

Paper 1 (Chapter 4) provided an introduction to augmentation for semantic segmentation, emphasizing the importance of domain-specific operations. For example, flipping an image of a river upside down could introduce meaningless information into the CNN and thus worsen the segmentation results. The paper further details image augmentation pipelines, including both offline and online approaches, and experimentally determines activation probabilities for the supported online augmentation operations on the River Water Segmentation Dataset (RIWA) [121]. These results were then used to compare offline augmentation with online augmentation, while simultaneously comparing 32 2D CNNs using the RIWA and Cityscapes (urban scene segmentation) [102] datasets.

In online augmentation, specific policies dictate the order of augmentation. Extensive testing on the RIWA has shown that geometric operations are slightly more beneficial than radiometric operations. The *geo first* policy, where images are rotated and/or resized before other geometric and finally radiometric operations are applied, has proven to be advantageous. This policy was also followed for the offline augmentation pipeline. Unlike online augmentation, the offline variant enables curation of all generated images.

The results showed that it is not necessary to experimentally determine online augmentation parameters that include activation probabilities, augmentation policies, and number of chained operations. With careful consideration of the domain and the use of AiSegs online augmentation preview function, these parameters can be set intuitively, producing results similar to those obtained using experimentally determined values.

When comparing models, the study found that the standard U-Net [122] with a ResNeXt backbone pre-trained on ImageNet outperformed others on the RIWA. However, the second-place Global

Convolutional Network (GCN) proved to be more computationally efficient, requiring only half the Video Random Access Memory (VRAM). In addition, offline augmentation was shown to be superior to online augmentation for all models. This is due to the patched input in online augmentation as opposed to full image augmentation followed by patch extraction in offline augmentation. However, the significant storage space requirements of offline augmentation make it impractical for larger datasets such as Cityscapes, where only online augmentation could prevent overfitting and improve model generalizability.

These findings were crucial in the dissertation, especially for augmenting 3D datasets, as discussed in Chapter 6, where the impracticality of offline augmentation due to large storage requirements necessitated online augmentation. Furthermore, both online and offline augmentation proved useful for augmenting synthetically generated training data, as synthetic data does not fully replicate the characteristics of real data (Chapter 6).

8.1.2 3D CNN Comparison

The logical consequence of the findings in Paper 1 (Chapter 4) would be to extend these pipelines to the third dimension. However, it proved insufficient to rely on augmentation alone to achieve optimal 3D segmentation results. Therefore, a prior step was to identify the most accurate 3D CNNs for carbon roving and pore segmentation.

Due to the computational requirements of large-scale comparisons of 3D CNNs, there have been no comprehensive evaluations on different datasets under identical training conditions. Paper 2 (Chapter 5) filled this gap by evaluating eight 3D CNN models on six datasets. However, since neural networks exhibit domain specificity, no general conclusions can be drawn from comparing only six datasets. Instead, due to the different characteristics of the datasets, the study provided tailored recommendations for the selection of a 3D CNN based on dataset characteristics.

The analysis showed that the 3D U-Net and its residual variant performed consistently well, and also achieved the best results in four datasets under the same training conditions. In particular, for the roving and pore segmentation tasks, the standard 3D U-Net provided the highest DICE score and therefore became the model of choice for Paper 3 (Chapter 6).

8.1.3 3D Data Augmentation for the Segmentation of Carbon Rovings

In paper 3 (Chapter 6), the findings from prior studies were integrated, adapting the online augmentation pipelines to 3D and employing the 3D U-Net for segmenting carbon from Computed Tomography (CT) scans of carbon reinforced concrete.

This study conducted a smaller comparison of four training scenarios with 3D augmentation: A training without augmentation to generate a baseline, a training with weak offline augmentation (only 3D rotations), a training with offline augmentation, and a training using online augmentation in combination with offline applied rotations around the X- and Y-axis.

The choice of combining offline rotation and online augmentation was driven by the constrained input dimensions of 128 x 128 x 64 voxels (VRAM limit did not allow a cubed input 128^3). When rotating a volume around the X- or Y-axis during online augmentation, it was anticipated that this causes significant mirroring effects, potentially impacting the results negatively.

The results showed that both offline and online augmentation performed satisfactorily on volumes containing the same type of roving as the training dataset. Offline augmentation resulted in fewer artifacts, while online augmentation resulted in numerous missegmentations that required post-processing. However, when a different type of roving was encountered, only online augmentation could segment it effectively, as the variance achievable with online augmentation is higher than with offline augmentation.

The subsequent discussion revealed that the dataset (training and validation data) lacked diversity. It was found that the validation data closely mirrored the characteristics of the training data, and thus cannot be used to assess overfitting. In addition, the online augmentation that was required to segment the second carbon type resulted in artifacts, suggesting that some augmentations were too aggressive. Another limitation was the need to resize the investigated volume by a scale factor of 0.5 to segment the second grid, which compromised segmentation accuracy at the roving-concrete transition zone.

8.1.4 Synthetic Training Data Generation

The study in Paper 3 (Chapter 6) emphasized the need for a larger, more variant dataset. Creating such a dataset manually requires significant labor. Consequently, Chapter 7 delved into the realm of deep learning-based augmentation using CUT, a variant of a Generative Adversarial Network (GAN). The model was designed to retain input shapes in the generalized images of the target domain. This property allowed CUT to be used for the purposes of this work: CUT as a generator of synthetic training data for image segmentation. By combining the GAN with augmentation techniques from Chapter 4, semi-supervised and unsupervised end-to-end trainings were performed using the GCN with a ResNeXt-50 backbone.

A key objective was to investigate the necessary data preparation steps for the GAN, in particular the composition of input images for the generation of synthetic CT images. Simple pore segmentation was chosen for investigation in order to assess the feasibility of a seemingly straightforward task. Two CT scans were selected as target domains: one of a predominantly homogeneous concrete with spherical pores, and another of a water reduced Strain-Hardening Cement-based Composite (SHCC) with spherical and non-spherical pores.

The results showed that input images consisting only of binary masks or masks from multi-class segmentation were insufficient. Introducing some structure, such as simple noise, into the input images significantly improved visual outcomes and mitigated GAN hallucinations.

Overall, successful generation of training data for pore segmentation was achieved for the two CT scans, with the SHCC data being significantly more challenging. The GCN was then trained on the synthetic datasets and segmentation of the two CT scans was performed. While the segmentation of the concrete CT scan with spherical pores was successful with minor errors, the task of pore segmentation in SHCC showed more errors due to the imperfect data generation by CUT.

Nevertheless, this chapter has demonstrated a successful application of unpaired image-to-image translation for synthetic training data generation in 2D. However, a stable implementation of 3D-CUT could not yet be achieved after initial tests.

8.2 Future Developments

Based on the results presented, further research is needed in the field of data augmentation and training data generation for CT data. Possible further areas of investigation are presented below.

8.2.1 Augmentation

Reducing Artifacts in Online Augmentation

The first paper (Chapter 4) suggested the benefits of online augmentation by augmenting the entire image and then randomly selecting a subsection for CNN input. After augmentation, the resulting image or volume may be smaller than the expected input size due to transformations such as squeezing or rotating, requiring mirroring of the image content to prevent unwanted resizing. Mirroring can introduce artifacts such as upside-down boats (Figure 8.1a and b), which are undesirable. Full images can be loaded alternatively. However, in the context of 3D augmentation, especially with volumes up to 3000 x 3000 x 3000 voxels, this approach becomes impractical. Instead, subsets of images or volumes that are slightly larger than the expected CNN input dimensions should be loaded for online augmentation, which cam help mitigate these artifacts (Figure 8.1c and b).

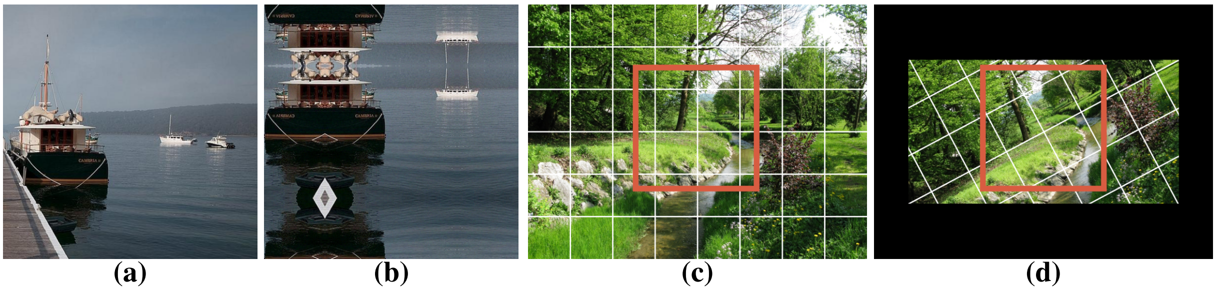


Figure 8.1: Effects of augmentations on the image content. When augmenting an image patch **(a)**, artifacts can occur due to mirroring **(b)**. Using larger input images before augmentation **(c)** and then cropping **(d)** can mitigate mirror artifacts.

Dataset Specific Offline Augmentation

While it was mentioned in paper 1 that online augmentation can be improved by augmenting the entire image (Section 4.4.3), the underlying problem of a dataset bias towards certain scenes was not further discussed. For example, if a CNN was trained on 10 surveillance images, each with dimensions of 500 x 500 pixels, and only one drone image with dimensions of 4000 x 4000 pixels, the CNN would have significantly fewer pixels associated with the surveillance scenes (2.5 megapixels) compared to the single drone image (16 megapixels). This discrepancy could heavily bias the CNN towards the drone footage, potentially resulting in higher performance on such images.

To identify potential biases of the RIWA, images were clustered into megapixel ranges with a step size of 0.25 in Figure 8.2a, revealing a peak of images with 1.75 to 2.0 megapixels. Approximately 30 % (Figure 8.2b) of all pixels fell within that range, with 150 of these images being drone images (16.7 % of all training pixels), potentially skewing the results.

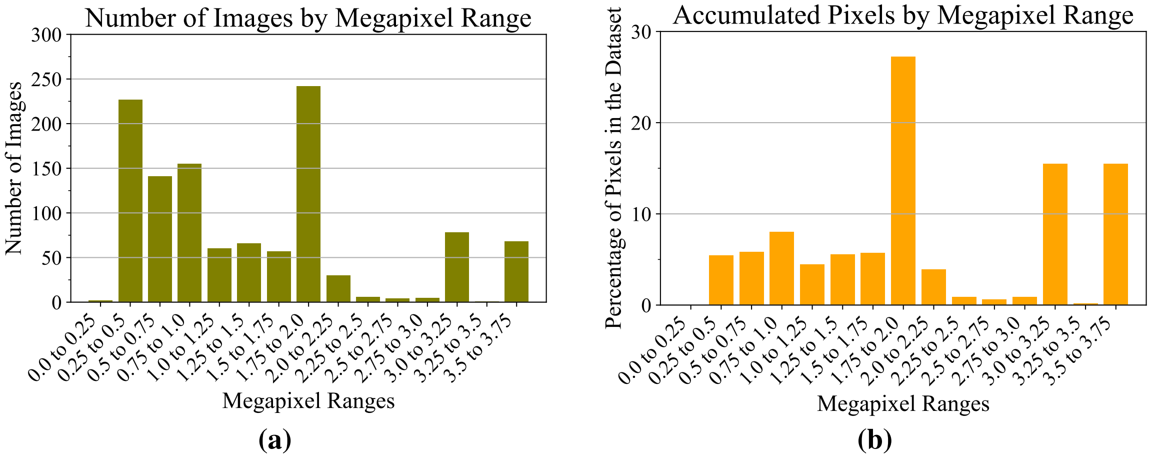


Figure 8.2: **(a)**: Number of images per 0.25 megapixels in the RIWA [121]. **(b)**: Percentage of accumulated pixels by megapixel range.

However, it's not entirely clear if this percentage alone leads to a bias, as more than 80 % of the images depict different scenes. Therefore, a further step is needed to cluster images based on their similarity, potentially using a neural network. Similar to the approach of the Fréchet Inception Distance in Chapter 7, a pretrained image classification network can serve as a feature extractor. For instance, using an Inception-v3 [107] or a more modern ResNet [123] pretrained on ImageNet and taking an earlier layer's output as a vector describing an image allows for comparison with others. This approach can help evaluate the dataset's variance and identify potential biases towards specific scenes.

Based on these analyses a potential solution is to first identify clusters, and sum the pixels contained in each cluster. By the use of offline augmentation, each cluster can then be extended so that all scenes are almost equally represented.

In the Pores Segmentation Dataset [46], the problem of the uneven distribution of voxels representing a scene was already taken into account during creation, but not accurately performed: to prevent a bias with regard to a specific concrete type, the training volumes of all CT scans were selected so that they contain approximately the same number of voxels. However, some CT volumes were smaller, leading to an underrepresentation and possibly making the segmentation of pores less accurate in certain volumes. In order to measure similarities, a 3D classification network would be required, which has not yet been published.

8.2.2 Pre-trained 3D Encoder

The results of Paper 1 (Chapter 4) highlighted the superior performance of networks with a pre-trained encoder compared to those without. While PyTorch readily accommodates pre-trained neural networks for 2D images on ImageNet [124], pre-trained 3D networks are lacking due to missing datasets.

The second paper (Chapter 5) presents several datasets and focuses on training neural networks for volume segmentation. These datasets can serve as valuable resources for pre-training a 3D encoder. However, to facilitate broader applicability, the 3D CNN used must have an encoder part that can be extracted without further modification for use in alternative architectures. One such CNN, the Med3D [35], was used in this study. In the publication, the authors have pre-trained this network on a novel dataset with different modalities. It's worth noting, that the authors of Med3D trained a segmentation CNN that requires a decoder component, which significantly increases the number of parameters and thus, for the sake of pretraining an encoder only, would unnecessarily increase the training duration.

Given the well-established success of residual neural networks (ResNets) in 2D classification tasks, a 3D residual encoder emerges as an attractive architectural choice. The transition from 2D to 3D is relatively straightforward, as the necessary CNNs are already available in PyTorch. This allows for the seamless replacement of normalization and convolutional layers with their 3D counterparts.

Therefore, a logical step is to create a 3D dataset for multiclass classification, which not only helps to potentially improve the performance of 3D CNNs, but also takes advantage of the similarity measurement mentioned in Section 8.2.1.

To create such a dataset, each volume could, for example, be labeled with "pores", "carbon", "fibers", etc. This approach allows the use of existing datasets presented in Chapter 5 with data augmentation from Chapter 6 and thus enables a relatively simple realization.

8.2.3 On the Quality Control of Carbon Reinforced Concrete

In Chapter 6, only a single segmented roving could be used to simulate the material behavior in a tensile test. Although all results were used to estimate the positional accuracy, the main goal was to simulate a tensile test. Without an adaptation of the algorithm to extract a representative volume element, three of the four segmentations could not be used. In one case, this was only due to the fact that two layers of the carbon grid were used. In the other two cases, it was due to the fact that the dataset was not suitable for this particular task due to the amount of coating.

Initially, the training dataset was created to automatically investigate the bonding behavior between the carbon and the concrete of a carbon grid that did not contain much coating (Figure 8.3a). It was only after analyzing the second type of carbon grids (Figure 8.3b) that it became clear that much more impregnation was used here, and that the diameter of the carbon would therefore be incorrectly determined, and the simulation would therefore yield incorrect results.

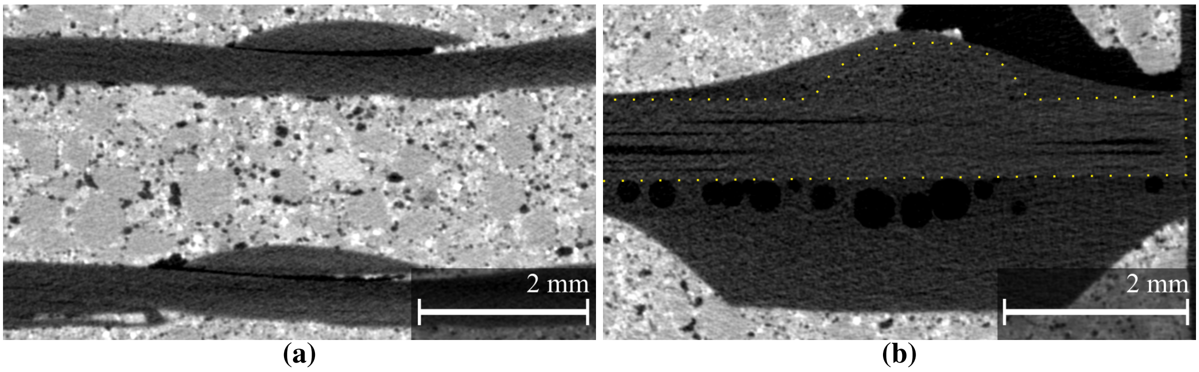


Figure 8.3: Two slices of CT scans of carbon reinforced concrete. Both slices show the intersection of two carbon rovings of a carbon grid. **(a)**: Carbon roving with almost no coating (SITgrid044 VL by Wilhelm Kneitz Solutions in Textile GmbH). **(b)**: Carbon roving with almost much coating (GRID Q43-CCE-21-E5 by solidian GmbH). The Carbon roving is marked with yellow dots.

A solution could be to either adjust the segmentation process or the carbon fiber strength parameter of the simulation. As shown in Figure 8.3, distinguishing between carbon and coating is extremely challenging, making manual generation of a 3D training dataset very laborious. Even using an unsupervised approach, as discussed in Chapter 7, to generate carbon without coating is prone to failure due to potential hallucinations by the GAN. Consequently, creating a dataset to extract the roving from the coating as well may not be justified due to the immense effort. Therefore, the simpler solution is to adjust the carbon fiber strength parameter based on the roving thickness.

In addition, achieving a perfect segmentation of the carbon introduces a problem for the determination of the concrete cover. This problem is illustrated in Figure 8.3b, where the coating extends nearly 2.0 mm below the carbon intersection (marked by yellow dots). As explained in the introduction (Section 1.2.1), where the accuracy requirements for carbon segmentation in concrete were discussed, solidian GmbH specify a minimum concrete coverage of 5 mm and a positional accuracy of ±0.2 mm [22]. In the specific investigation of paper 3 (Chapter 6), removing the coating from the segmentation would exhaust the required positional accuracy.

However, the CNN can still be improved in terms of generalizability to segment carbon and coating from concrete: extending the training data will lead to reduced segmentation artifacts. To achieve this goal, a 3D version of CUT can be used. As mentioned in Chapter 7, by scanning a carbon roving without concrete, the GAN only needs to generate the concrete surrounding the roving. This simplifies dataset generation by eliminating the need to manually model constituents. The carbon fiber strength parameter can then be adjusted based on the roving thickness.

8.2.4 Subvoxel Accurate Segmentation

Precise subvoxel segmentation in tomography data offers many benefits, with shorter CT scan times and more accurate estimation of the segmented volume being the most important aspects.

A CT scan of reinforced concrete typically takes about 1 hour and 40 minutes at a voxel size of 10 µm and can be reduced to as little as 40 minutes at a voxel size of 20 µm (using the Procon CT-XPRESS [86]). This difference is due to the geometry of a cone beam CT and the relative position of an object between the radiation source and the detector (Figure 8.4). When an object (4) is closer to the X-ray source (1), it occupies more space on the detector (6) and thus in each projection. This means that a small rotation step size is required so that a point on the object surface does not move too far in two successive projections (from position 2 to 3), causing a loss of detail. Therefore, when using larger voxel sizes, the object is closer (5) to the detector and larger rotation steps can be used, thus reducing the scan time. This is especially important when performing hundreds of CT scans. Subvoxel accuracy can result in nearly similarly accurate segmentations when scanning with larger voxel sizes compared to standard voxel-wise segmentation and smaller voxel sizes. Depending on the required segmentation accuracy, subvoxel accurate segmentation can be a valuable tool to reduce scan time.

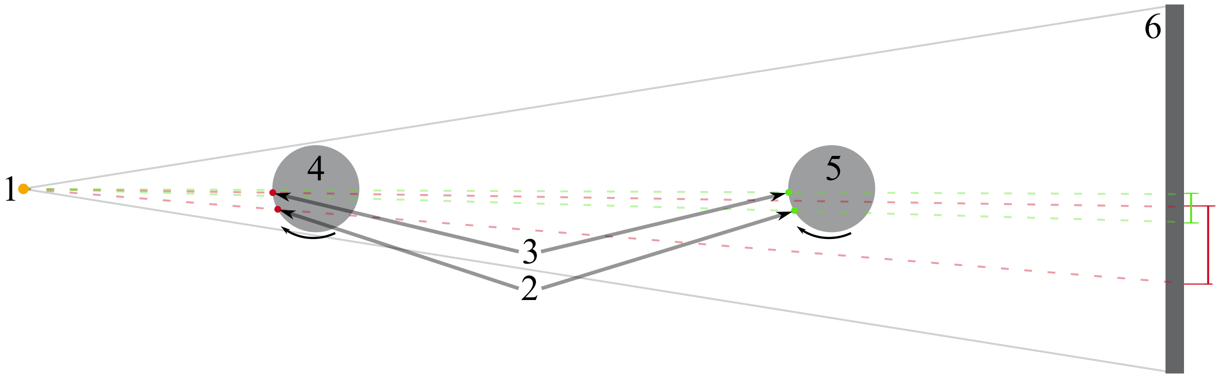


Figure 8.4: Geometry in a cone beam CT scanner and the effect of the same rotation step size at two object positions. 1: X-ray source. 2 and 3: A point on the surface of the object before and after rotation. 4 and 5: The same object closer and further away from the X-ray source. 6: Detector.
The red and green lines mark the relative positions of the moved surface points projected onto the detector (before (2) and after (3) a rotation). At position 4, the distance of the same point to the detector after a rotation is greater (red mark at 6) than at position 5 (red mark at 5).

When determining porosity in a volume, the scan time cannot be reduced because the accuracy of the pore segmentation is important for determining pore volumes. It is also not possible to move the object closer to the radiation source, since the size of the detector is fixed and an object would potentially be moved out of the projection image. Therefore, subvoxel accurate segmentation would be advantageous for a more accurate result. For example, when estimating the volume of a sphere, the radius is cubed ($V = \frac{3}{4}\pi r^3$). In a CT scan of concrete, a subvoxel accurate radius will increase the accuracy of the porosity determination.

However, a CNN that can segment with subvoxel accuracy requires a slightly different architecture: The decoder part must be extended by one level. This has already been performed in 2D [125, 126] and is called semantic segmentation super-resolution [127], but 3D versions have not yet been published.

For training, the input volumes would have a lower resolution than the target masks. Such a dataset can be generated using 3D CUT, which has shown its potential for pore generation in 2D. By generating larger pores, the generated images can then be downsampled by a user-defined scale, e.g. 4x. The downsampled images are now the input and the high-resolution masks are the target for training a CNN for semantic segmentation.

Since a 3D CUT is not yet available, non-machine learning methods may be used to create the dataset. In this case, an existing target volume of a semantic segmentation training dataset would be upsampled (e.g., 4x) and represent the target. However, deep learning approaches have significantly outperformed these since 2014 (SRCNN) [128] and have also been investigated in 3D [129] and may therefore be more suitable for volume upsampling.

8.2.5 Towards Volume-to-Volume Translation

In Chapter 7, a 2D reimplementation of CUT was used to generate synthetic training data for semantic segmentation. As already mentioned, it has not yet been possible to implement a stable 3D version. Therefore, this unstable behavior needs to be investigated and solved. There are several ways to adapt the GAN. These include adjusting the number of noise contrastive estimation patches, using label noise, weighting a loss more than another, adapting architectures and other hyperparameters, or implementing a discriminator learning rate threshold. This can be necessary because the 3D discriminator currently outperforms the generator, and training is unstable, sometimes resulting in catastrophic forgetting.

8.3 Conclusion

In this dissertation, different approaches to improve the segmentation of tomography data using neural networks were investigated, focusing on both supervised and unsupervised methods. Throughout the research, two developed softwares played an essential role in facilitating the studies: AiSeg and unpAIred. AiSeg, was used in all stages of the investigations, providing support for data pre-processing, augmentation, analysis and application of neural networks. Specifically designed for the domain translation study, unpAIred enabled the use of deep learning-based augmentation using Contrastive Unpaired Translation (CUT) to generate synthetic training data.

The research results collectively advance the field of voxel data segmentation through the use of augmentation techniques and deep learning methods. Augmentation pipelines have been explored, with an emphasis on domain-specific operations that are critical for semantic segmentation tasks. Careful selection and ordering of augmentation operations significantly affected model performance, with insights gained from comparing different CNN architectures and augmentation strategies on different datasets.

An evaluation of 3D CNN models on different datasets identified the most accurate models for carbon roving and pore segmentation and provided recommendations for selecting appropriate architectures based on dataset characteristics. This comparative analysis laid the foundation for subsequent segmentation tasks.

Integrating the findings from augmentation and 3D CNN comparison, augmentation pipelines were extended to 3D and applied to segment carbon from CT scans of carbon reinforced concrete. The analysis of different augmentation strategies demonstrated the effectiveness online augmentation, while highlighting challenges such as the need for diverse training data and careful parameter tuning to mitigate segmentation artifacts.

To address the need for larger and more diverse datasets, research into using CUT to generate synthetic training data has shown promising results, despite challenges with data preparation and model stability in 3D. This approach offers potential for semi-supervised training methods and addresses the limitations of manual dataset generation.

Overall, the research contributes to improving the performance of segmentation models while addressing the challenges of data scarcity through synthetic data generation and augmentation. Further exploration of advanced augmentation strategies, model architectures and subvoxel segmentation holds promise for advancing the segmentation of complex tomography data.

References

[1] A. Brand et al. "Beyond authorship: attribution, contribution, collaboration, and credit." In: *Learned Publishing* 28.2 (2015), pp. 151–155. DOI: 10.1087/20150211.

[2] F. Wagner, A. Eltner, and H.-G. Maas. "River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs." In: *Int. J. Appl. Earth Obs. Geoinf.* 119 (2023), p. 103305. DOI: 10.1016/j.jag.2023.103305.

[3] F. Wagner and H.-G. Maas. "A Comparative Study of Deep Architectures for Voxel Segmentation in Volume Images." In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVIII-1/W2-2023 (2023), pp. 1667–1676. DOI: 10.5194/isprs-archives-XLVIII-1-W2-2023-1667-2023.

[4] F. Wagner et al. "Analysis of Thin Carbon Reinforced Concrete Structures through Microtomography and Machine Learning." In: *Buildings* 13.9 (2023). DOI: 10.3390/buildings13092399.

[5] T. Park et al. "Contrastive Learning for Unpaired Image-to-Image Translation." In: *CoRR* abs/2007.15651 (2020). DOI: 10.48550/arXiv.2007.15651.

[6] B. Beckmann et al. "Collaborative research on carbon reinforced concrete structures in the CRC/TRR 280 project." In: *Civil Engineering Design* 3.3 (2021), pp. 99–109. DOI: 10.1002/cend.202100017.

[7] I. Vakaliuk, S. Scheerer, and M. Curbach. "Initial laboratory test of load-bearing shell-shaped TRC structures." In: *Concrete Innovation for Sustainability*. June 2022, pp. 675–684.

[8] I. Vakaliuk, S. Scheerer, and M. Curbach. "Vacuum-Assisted Die Casting Method for the Production of Filigree Textile-Reinforced Concrete Structures." In: *Buildings* 13.10 (2023). DOI: 10.3390/buildings13102641.

[9] A. Badran et al. "Automated segmentation of computed tomography images of fiber-reinforced composites by deep learning." In: *Journal of Materials Science* 55.34 (Dec. 2020), pp. 16273–16289. DOI: 10.1007/s10853-020-05148-7.

[10] P. Iassonov, T. Gebrenegus, and M. Tuller. "Segmentation of X-ray computed tomography images of porous materials: A crucial step for characterization and quantitative analysis of pore structures." In: *Water Resources Research* 45.9 (2009). DOI: 10.1029/2009WR008087.

[11] Y. Dong et al. "Microstructural crack segmentation of three-dimensional concrete images based on deep convolutional neural networks." In: *Construction and Building Materials* 253 (2020), p. 119185. DOI: 10.1016/j.conbuildmat.2020.119185.

[12] F. B. Salling et al. "Individual fibre inclination segmentation from X-ray computed tomography using principal component analysis." In: *Journal of Composite Materials* 56.1 (2022), pp. 83–98. DOI: 10.1177/00219983211052741.

[13] L. Hong et al. "Effective segmentation of short fibers in glass fiber reinforced concrete's X-ray images using deep learning technology." In: *Materials & Design* 210 (2021), p. 110024. DOI: 10.1016/j.matdes.2021.110024.

[14] C. Jung et al. "Towards automatic crack segmentation in 3d concrete images." In: *11th Conference on Industrial Computed Tomography, Wels, Austria (iCT 2022)*. Vol. 27. 3. Feb. 2022. DOI: 10.58286/26620.

[15] S. Bellens et al. "Evaluating conventional and deep learning segmentation for fast X-ray CT porosity measurements of polymer laser sintered AM parts." In: *Polymer Testing* 110 (2022), p. 107540. DOI: 10.1016/j.polymertesting.2022.107540.

[16] A. Plaksyvyi, M. Skublewska-Paszkowska, and P. Powroznik. "A Comparative Analysis of Image Segmentation Using Classical and Deep Learning Approach." In: *Advances in Science and Technology Research Journal* 17.6 (2023), pp. 127–139. DOI: 10.12913/22998624/172771.

[17] A. Işın, C. Direkoğlu, and M. Şah. "Review of MRI-based Brain Tumor Image Segmentation Using Deep Learning Methods." In: *Procedia Computer Science* 102 (2016). 12th International Conference on Application of Fuzzy Systems and Soft Computing, ICAFS 2016, 29-30 August 2016, Vienna, Austria, pp. 317–324. DOI: 10.1016/j.procs.2016.09.407.

[18] A. Garcia-Garcia et al. "A survey on deep learning techniques for image and video semantic segmentation." In: *Applied Soft Computing* 70 (2018), pp. 41–65. DOI: 10.1016/j.asoc.2018.05.018.

[19] S. Thiruchittampalam et al. "Comparative Evaluation of Traditional and Deep Learning-Based Segmentation Methods for Spoil Pile Delineation Using UAV Images." In: (2024). DOI: 10.48550/arXiv.2402.00295.

[20] M. Reinhardt et al. "Benchmarking conventional and machine learning segmentation techniques for digital rock physics analysis of fractured rocks." In: *Environmental Earth Sciences* 81.3 (Jan. 2022), p. 71. DOI: 10.1007/s12665-021-10133-7.

[21] S. Moccia et al. "Blood vessel segmentation algorithms — Review of methods, datasets and evaluation metrics." In: *Computer Methods and Programs in Biomedicine* 158 (2018), pp. 71–91. DOI: 10.1016/j.cmpb.2018.02.001.

[22] A. Shams. *Hochleistungsbewehrungen für dünne Betonbauteile*. PDF document. https://www.dafstb.de/application/fachkolloquien/2017/Anlage13_2017-04-20_Hochleistungsbewehrungen_duenne_Betonbauteile_Shams.pdf (Accessed on 06.04.2024). solidian GmbH, 2017.

[23] M. Kupke. "CUBE Projektvorstellung." In: *Beton- und Stahlbetonbau* 118.S2 (2023), pp. 13–21. DOI: 10.1002/best.202200021.

[24] *green building nachhaltig mit Beton bauen*. PDF document. https://solidian.com/wp-content/uploads/solidian_Green_Building_2023_DE_web.pdf (Accessed on 06.04.2024). solidian GmbH, 2023.

[25] F. Liebold et al. "Damage Analysis and Quality Control of Carbon-Reinforced Concrete Beams Based on In Situ Computed Tomography Tests." In: *Buildings* 13.10 (2023). DOI: 10.3390/buildings13102669.

[26] M. A. Vicente et al. "Use of Computed Tomography Scan Technology to Explore the Porosity of Concrete: Scientific Possibilities and Technological Limitations." In: *Applied Sciences* 11.18 (2021). DOI: 10.3390/app11188699.

[27] C. L. Reedy and C. L. Reedy. "High-resolution micro-CT with 3D image analysis for porosity characterization of historic bricks." In: *Heritage Science* 10.1 (June 2022), p. 83. DOI: 10.1186/s40494-022-00723-4.

[28] T. Barisin et al. "Methods for segmenting cracks in 3d images of concrete: A comparison based on semi-synthetic images." In: *Pattern Recognition* 129 (2022), p. 108747. DOI: 10.1016/j.patcog.2022.108747.

[29] C. Jung et al. "3d imaging and analysis of cracks in loaded concrete samples." In: *12th Conference on Industrial Computed Tomography, Fürth, Germany (iCT 2023)*. Vol. 28. 3. Mar. 2023. DOI: 10.58286/27721.

[30] T. Barisin et al. "Crack Segmentation in 3D Concrete Images: Perspectives and Challenges." In: *e-Journal of Nondestructive Testing* 27.9 (Aug. 2022). International Symposium on Non-Destructive Testing in Civil Engineering. DOI: 10.58286/27207.

[31] K. Ehrig et al. "Comparison of Crack Detection Methods for Analyzing Damage Processes in Concrete with Computed Tomography." In: *International Symposium on Digital Industrial Radiology and Computed Tomography*. Berlin, Germany, July 2011.

[32] Ö. Çiçek et al. "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation." In: *CoRR* abs/1606.06650 (2016). DOI: 10.48550/arXiv.1606.06650.

[33] L. Yu et al. "Automatic 3D Cardiovascular MR Segmentation with Densely-Connected Volumetric ConvNets." In: *CoRR* abs/1708.00573 (2017). DOI: 10.48550/arXiv.1708.00573.

[34] W. Li et al. "On the Compactness, Efficiency, and Representation of 3D Convolutional Networks: Brain Parcellation as a Pretext Task." In: (2017). Ed. by M. Niethammer et al., pp. 348–360. DOI: 10.1007/978-3-319-59050-9_28.

[35] S. Chen, K. Ma, and Y. Zheng. "Med3D: Transfer Learning for 3D Medical Image Analysis." In: *CoRR* abs/1904.00625 (2019). DOI: 10.48550/arXiv.1904.00625.

[36] K. Lee et al. "Superhuman Accuracy on the SNEMI3D Connectomics Challenge." In: *CoRR* abs/1706.00120 (2017). DOI: 10.48550/arXiv.1706.00120.

[37] T. D. Bui, J. Shin, and T. Moon. "Skip-connected 3D DenseNet for volumetric infant brain MRI segmentation." In: *Biomedical Signal Processing and Control* 54 (2019), p. 101613. DOI: 10.1016/j.bspc.2019.101613.

[38] F. Milletari, N. Navab, and S. Ahmadi. "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation." In: *CoRR* abs/1606.04797 (2016). DOI: 10.48550/arXiv.1606.04797.

[39] T. Lei et al. "Lightweight V-Net for Liver Segmentation." In: (2020), pp. 1379–1383. DOI: 10.1109/ICASSP40776.2020.9053454.

[40] S. P. Singh et al. "3D Deep Learning on Medical Images: A Review." In: (2020). DOI: 10.48550/ARXIV.2004.00218.

[41] B. H. Menze et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)." In: *IEEE Trans. Med. Imaging* 34.10 (Oct. 2015), pp. 1993–2024. DOI: 10.1109/TMI.2014.2377694.

[42] J.-Y. Zhu et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.

[43] Y. He et al. "Deep Learning based 3D Segmentation: A Survey." In: *CoRR* abs/2103.05423 (2021). DOI: 10.48550/arXiv.2103.05423.

[44] F. Wagner. "Fiber Segmentation Dataset." In: (2023). DOI: 10.34740/KAGGLE/DS/2894881.

[45] F. Wagner. "Carbon Rovings Segmentation Dataset." In: (2023). DOI: 10.34740/KAGGLE/DS/2920892.

[46] F. Wagner. "Concrete Pores Segmentation Dataset." In: (2023). DOI: 10.34740/KAGGLE/DS/2921245.

[47] R. Strudel et al. "Segmenter: Transformer for Semantic Segmentation." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2021, pp. 7242–7252. DOI: 10.1109/ICCV48922.2021.00717.

[48] A. Hatamizadeh et al. "UNETR: Transformers for 3D Medical Image Segmentation." In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 1748–1758. DOI: 10.1109/WACV51458.2022.00181.

[49] A. Hatamizadeh et al. "Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images." In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Ed. by A. Crimi and S. Bakas. Cham: Springer International Publishing, 2022, pp. 272–284. DOI: 10.1007/978-3-031-08999-2_22.

[50] F. Yuan, Z. Zhang, and Z. Fang. “An effective CNN and Transformer complementary network for medical image segmentation.” In: *Pattern Recognition* 136 (2023), p. 109228. DOI: 10.1016/j.patcog.2022.109228.

[51] H.-Y. Zhou et al. “nnFormer: Volumetric Medical Image Segmentation via a 3D Transformer.” In: *IEEE Transactions on Image Processing* 32 (2023), pp. 4036–4045. DOI: 10.1109/TIP.2023.3293771.

[52] J. Maurício, I. Domingues, and J. Bernardino. “Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review.” In: *Applied Sciences* 13.9 (2023). DOI: 10.3390/app13095521.

[53] S. L. Smith et al. *ConvNets Match Vision Transformers at Scale*. 2023. DOI: 10.48550/arXiv.2310.16764.

[54] A. Brock et al. “High-Performance Large-Scale Image Recognition Without Normalization.” In: *CoRR* abs/2102.06171 (2021). DOI: 10.48550/arXiv.2102.06171.

[55] L. Perez and J. Wang. “The Effectiveness of Data Augmentation in Image Classification using Deep Learning.” In: *CoRR* abs/1712.04621 (2017). DOI: 10.48550/arXiv.1712.04621.

[56] A. Mumuni and F. Mumuni. “Data augmentation: A comprehensive survey of modern approaches.” In: *Array* 16 (2022), p. 100258. DOI: 10.1016/j.array.2022.100258.

[57] S. Yang et al. *Image Data Augmentation for Deep Learning: A Survey*. 2023. DOI: 10.48550/arXiv.2204.08610.

[58] K. Alomar, H. I. Aysel, and X. Cai. “Data augmentation in classification and segmentation: A survey and new strategies.” In: *Journal of Imaging* 9.2 (Feb. 2023). DOI: 10.3390/jimaging9020046.

[59] C. Shorten and T. M. Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning.” In: *Journal of Big Data* 6.1 (2019), p. 60. DOI: 10.1186/s40537-019-0197-0.

[60] A. Buslaev et al. “Albumentations: Fast and Flexible Image Augmentations.” In: *Information* 11.2 (Feb. 2020), p. 125. DOI: 10.3390/info11020125.

[61] T. Qin et al. “Automatic Data Augmentation Via Deep Reinforcement Learning for Effective Kidney Tumor Segmentation.” In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 1419–1423. DOI: 10.1109/ICASSP40776.2020.9053403.

[62] X. Huang et al. “Multimodal Unsupervised Image-to-Image Translation.” In: *CoRR* abs/1804.04732 (2018). DOI: 10.48550/arXiv.1804.04732.

[63] B. T. Imbusch, M. Schwarz, and S. Behnke. “Synthetic-to-Real Domain Adaptation using Contrastive Unpaired Translation.” In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. 2022, pp. 595–602. DOI: 10.1109/CASE49997.2022.9926640.

[64] J. Myers et al. “Modified CycleGAN for the synthesization of samples for wheat head segmentation.” In: (2024). DOI: 10.48550/arXiv.2402.15135.

[65] Y. AlNoamany and J. A. Borghi. “Towards computational reproducibility: researcher perspectives on the use and sharing of software.” In: *PeerJ Computer Science* 4 (Sept. 2018), e163. DOI: 10.7717/peerj-cs.163.

[66] R. Pruim, M.-C. Gîrjău, and N. J. Horton. “Fostering Better Coding Practices for Data Scientists.” In: *Harvard Data Science Review* 5.3 (July 2023). https://hdsr.mitpress.mit.edu/pub/8wsiqh1c. DOI: 10.1162/99608f92.97c9f60f.

[67] Django Software Foundation. *Django*. Version 4.2.7. https://djangoproject.com. Nov. 1, 2023.

[68] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In: (2019). Ed. by H. Wallach et al., pp. 8024–8035. DOI: 10.48550/arXiv.1912.01703.

[69] C. R. Harris et al. "Array programming with NumPy." In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[70] Y. Liu et al. *PaddleSeg: A High-Efficient Development Toolkit for Image Segmentation.* 2021. DOI: 10.48550/arXiv.2101.06175.

[71] H. Hunter-Zinck et al. "Ten simple rules on writing clean and reliable open-source scientific software." In: *PLOS Computational Biology* 17.11 (Nov. 2021), pp. 1–9. DOI: 10.1371/journal.pcbi.1009481.

[72] G. van Rossum, B. Warsaw, and N. Coghlan. *Style Guide for Python Code.* PEP 8. https://www.python.org/dev/peps/pep-0008/ (Accessed on 28.04.2024). 2001.

[73] JetBrains. *PyCharm.* Version 2023.3.5. Mar. 21, 2024.

[74] P. Runeson. "A survey of unit testing practices." In: *IEEE Software* 23.4 (2006), pp. 22–29. DOI: 10.1109/MS.2006.91.

[75] Z. A. Aziz et al. "Python Parallel Processing and Multiprocessing: A Review." In: *Academic Journal of Nawroz University* (2021). DOI: 10.25007/ajnu.v10n3a1145.

[76] S. Zheng et al. "Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers." In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* Los Alamitos, CA, USA: IEEE Computer Society, June 2021, pp. 6877–6886. DOI: 10.1109/CVPR46437.2021.00681.

[77] X. Li et al. "Transformer-Based Visual Segmentation: A Survey." In: *ArXiv* abs/2304.09854 (2023). DOI: 10.48550/arXiv.2304.09854.

[78] S. Prabhu et al. "Production of X-RAYS using X-RAY Tube." In: *Journal of Physics: Conference Series* 1712.1 (Dec. 2020), p. 012036. DOI: 10.1088/1742-6596/1712/1/012036.

[79] N. Tesla. "High frequency oscillators for electro-therapeutic and other purposes." In: *Proceedings of the IEEE* 87.7 (1999), pp. 1282–. DOI: 10.1109/JPROC.1999.771079.

[80] A. Sommerfeld. "Über die Verteilung der Intensität bei der Emission von *Röntgen*strahlen." German. In: *Phys. Z.* 10 (1909), pp. 969–976.

[81] H. A. Kramers. "XCIII. On the theory of X-ray absorption and of the continuous X-ray spectrum." In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 46.275 (1923), pp. 836–871. DOI: 10.1080/14786442308565244.

[82] Y. Tan et al. "Beam hardening correction and its influence on the measurement accuracy and repeatability for CT dimensional metrology applications." In: *4th Conference on Industrial Computed Tomography (iCT), 19-21 September 2012, Wels, Austria.* Vol. 17. 2012, pp. 355–362.

[83] MITOS GmbH Munich, Germany. *X-AID.* Version 2024.2.1. Feb. 1, 2024.

[84] J. A. Seibert and J. M. Boone. "X-Ray Imaging Physics for Nuclear Medicine Technologists. Part 2: X-Ray Interactions and Image Formation." In: *Journal of Nuclear Medicine Technology* 33.1 (2005), pp. 3–18.

[85] T. Buzug. *Computed Tomography. From Photon Statistics to Modern Cone-Beam CT.* Vol. 1. Published: 20 May 2008. Berlin, Heidelberg: Springer Berlin, Heidelberg, May 2008, pp. XIV, 522. DOI: 10.1007/978-3-540-39408-2.

[86] procon GmbH. *CT-XPRESS.* 2015. URL: http://www.pxr-gmbh.de/wp-content/uploads/2015/04/CT-XPRESS_EN.pdf (visited on 03/22/2024).

[87] V. I. Corporation. *Industrial 4343HE.* 2022. URL: https://www.vareximaging.com/wp-content/uploads/2022/01/4343HE_PDS_135979-000.pdf (visited on 03/22/2024).

[88] X. Ou et al. "Recent development in X-ray imaging technology: Future and challenges." In: *Research (Wash. D.C.)* 2021 (Dec. 2021). DOI: 10.34133/2021/9892152.

[89] M. Nikl. "Scintillation detectors for x-rays." In: *Measurement Science and Technology* 17.4 (Feb. 2006), R37. DOI: 10.1088/0957-0233/17/4/R01.

[90] M. Lüthi et al. "X-ray flat-panel detector geometry correction to improve dimensional computed tomography measurements." In: *Measurement Science and Technology* 31.3 (Dec. 2019), p. 035002. DOI: 10.1088/1361-6501/ab52b1.

[91] M. Ferrucci et al. "Towards geometrical calibration of x-ray computed tomography systems—a review." In: *Measurement Science and Technology* 26.9 (Aug. 2015), p. 092003. DOI: 10.1088/0957-0233/26/9/092003.

[92] V. Nguyen et al. "A low-cost geometry calibration procedure for a modular cone-beam X-ray CT system." In: *Nondestructive Testing and Evaluation* 35.3 (2020), pp. 252–265. DOI: 10.1080/10589759.2020.1774580.

[93] T. Rodet, F. Noo, and M. Defrise. "The cone-beam algorithm of Feldkamp, Davis, and Kress preserves oblique line integrals." In: *Medical Physics* 31.7 (2004), pp. 1972–1975. DOI: 10.1118/1.1759828.

[94] R. Schofield et al. "Image reconstruction: Part 1 – understanding filtered back projection, noise and image acquisition." In: *Journal of Cardiovascular Computed Tomography* 14.3 (2020), pp. 219–225. DOI: 10.1016/j.jcct.2019.04.008.

[95] L. Xue et al. "Numerical Analysis of the Feldkamp–Davis–Kress Effect on Industrial X-Ray Computed Tomography for Dimensional Metrology." In: *Journal of Computing and Information Science in Engineering* 15.2 (June 2015), p. 021008. DOI: 10.1115/1.4028942.

[96] C. Peng et al. "Large Kernel Matters - Improve Semantic Segmentation by Global Convolutional Network." In: *CoRR* abs/1703.02719 (2017). DOI: 10.48550/arXiv.1703.02719.

[97] S. Xie et al. "Aggregated Residual Transformations for Deep Neural Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5987–5995. DOI: 10.1109/CVPR.2017.634.

[98] K. Man and J. Chahl. "A Review of Synthetic Image Data and Its Use in Computer Vision." In: *Journal of Imaging* 8.11 (2022). DOI: 10.3390/jimaging8110310.

[99] M. Toldo et al. "Unsupervised Domain Adaptation in Semantic Segmentation: A Review." In: *Technologies* 8.2 (2020). DOI: 10.3390/technologies8020035.

[100] G. Csurka, R. Volpi, and B. Chidlovskii. "Unsupervised Domain Adaptation for Semantic Image Segmentation: a Comprehensive Survey." In: *CoRR* abs/2112.03241 (2021). DOI: 10.48550/arXiv.2112.03241.

[101] H. Lee et al. "Diverse Image-to-Image Translation via Disentangled Representations." In: *Computer Vision – ECCV 2018*. Ed. by V. Ferrari et al. Cham: Springer International Publishing, 2018, pp. 36–52. DOI: 10.1007/978-3-030-01246-5_3.

[102] M. Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. DOI: 10.48550/arXiv.1604.01685.

[103] I. J. Goodfellow et al. "Generative adversarial nets." In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 2672–2680.

[104] H. Thanh-Tung and T. Tran. "Catastrophic forgetting and mode collapse in GANs." In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–10. DOI: 10.1109/IJCNN48605.2020.9207181.

[105] M. Heusel et al. "GANs trained by a two time-scale update rule converge to a local nash equilibrium." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6629–6640.

[106] N. S. Detlefsen et al. "TorchMetrics - Measuring Reproducibility in PyTorch." In: *Journal of Open Source Software* 7.41 (2022), p. 101. DOI: 10.21105/joss.04101.

[107] C. Szegedy et al. "Rethinking the Inception Architecture for Computer Vision." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[108] M. Seitzer. *pytorch-fid: FID Score for PyTorch*. https://github.com/mseitzer/pytorch-fid. https://github.com/mseitzer/pytorch-fid (Version 0.3.0). Aug. 2020.

[109] M. Lucic et al. "Are GANs Created Equal? A Large-Scale Study." In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 698–707.

[110] T. Karras et al. "Training Generative Adversarial Networks with Limited Data." In: vol. abs/2006.06676. 2020. DOI: 10.48550/arXiv.2006.06676.

[111] K. You et al. "How Does Learning Rate Decay Help Modern Neural Networks?" In: *CoRR* abs/1908.01878 (2019). DOI: 10.48550/arXiv.1908.01878.

[112] I. Loshchilov and F. Hutter. "Fixing Weight Decay Regularization in Adam." In: *CoRR* abs/1711.05101 (2017). DOI: 10.48550/arXiv.1711.05101.

[113] Y. Wu and K. He. "Group Normalization." In: *CoRR* abs/1803.08494 (2018). DOI: 10.48550/arXiv.1803.08494.

[114] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: (2015). Ed. by Y. Bengio and Y. LeCun. DOI: 10.48550/arXiv.1412.6980.

[115] S. J. Reddi, S. Kale, and S. Kumar. "On the Convergence of Adam and Beyond." In: *CoRR* abs/1904.09237 (2019). DOI: 10.48550/arXiv.1904.09237.

[116] S. Loehnert et al. "An enriched phase-field method for the efficient simulation of fracture processes." In: *Computational Mechanics* 71.5 (May 2023), pp. 1015–1039.

[117] C. Krüger, V. Curoşu, and S. Loehnert. "An extended phase-field approach for the efficient simulation of fatigue fracture processes." In: *International Journal for Numerical Methods in Engineering* 125.7 (2024), e7422. DOI: 10.1002/nme.7422.

[118] X. Tian et al. "OCT2Confocal: 3D CycleGAN based Translation of Retinal OCT Images to Confocal Microscopy." In: *CoRR* abs/2311.10902 (2023). DOI: 10.48550/ARXIV.2311.10902.

[119] D. Abramian and A. Eklund. "Generating fMRI volumes from T1-weighted volumes using 3D CycleGAN." In: (2019). DOI: 10.48550/arXiv.1907.08533.

[120] G. Shafai-Erfani et al. "MRI-Based Proton Treatment Planning for Base of Skull Tumors." In: *International Journal of Particle Therapy* 6.2 (2019), pp. 12–25. DOI: 10.14338/IJPT-19-00062.1.

[121] X. Blanch, F. Wagner, and A. Eltner. *RIWA Dataset*. 2022. DOI: 10.34740/KAGGLE/DSV/4289421.

[122] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *CoRR* abs/1505.04597 (2015). DOI: 10.48550/arXiv.1505.04597.

[123] K. He et al. "Deep Residual Learning for Image Recognition." In: *CoRR* abs/1512.03385 (2015). DOI: 10.48550/arXiv.1512.03385.

[124] J. Deng et al. "Imagenet: A large-scale hierarchical image database." In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[125] Q. Delannoy et al. "SegSRGAN: Super-resolution and segmentation using generative adversarial networks — Application to neonatal brain MRI." In: *Computers in Biology and Medicine* 120 (2020), p. 103755. DOI: 10.1016/j.compbiomed.2020.103755.

[126] J. Jiang et al. "Super-resolution semantic segmentation with relation calibrating network." In: *Pattern Recognition* 124 (2022), p. 108501. DOI: 10.1016/j.patcog.2021.108501.

[127] L. Wang et al. "Dual Super-Resolution Learning for Semantic Segmentation." In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 3773–3782. DOI: `10.1109/CVPR42600.2020.00383`.

[128] C. Dong et al. "Image Super-Resolution Using Deep Convolutional Networks." In: *CoRR* abs/1501.00092 (2015). DOI: `10.48550/arXiv.1501.00092`.

[129] Y. Li et al. "VolumeNet: A Lightweight Parallel Network for Super-Resolution of MR and CT Volumetric Data." In: *IEEE Transactions on Image Processing* 30 (2021), pp. 4840–4854. DOI: `10.1109/TIP.2021.3076285`.

List of Tables

List of Figures

List of Abbreviations

µ-CT micro-Computed Tomography

BraTS Brain Tumor Segmentation Challenge

CNN Convolutional Neural Network

COCO Common Objects in Context

CP Concrete Pores

CRC Carbon Reinforced Concrete

CRUD Create, Read, Update, Delete

CSS Cascading Style Sheets

CT Computed Tomography

CUDA Compute Unified Device Architecture

CUT Contrastive Unpaired Translation

DRL Deep Reinforcement Learning

DSLR Digital Single-Lens Reflex

ED Edema

EM Electron Microscopy

ET Enhancing Tumor

FDK Feldkamp-David-Kress

FID Fréchet Inception Distance

FN False Negative

FP False Positives

GAN Generative Adversarial Network

GCN Global Convolutional Network

GD Gadolinium

GPU Graphics Processing Unit

GT Ground Truth

GUI Graphical User Interface

HP Hessian-based Percolatio

HPC High Performance Computing

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

LabMorTex Laboratory Mortar Extruder

MLP Multilayer Perceptron

MRI Magnetic Resonance Imaging

NCR Necrotic Tumor core

NET Non-Enhancing Tumor core

NFNet Normalizer-Free Network

NURBS Non-Uniform Rational B-Splines

PEP Python Enhancement Proposal

RAM Random Access Memory

RIWA River Water Segmentation Dataset

RVE Representative Volume Element

SBIGA Scaled Boundary Isogeometric Analysis

SHCC Strain-Hardening Cement-based Composite

Synth-CP Synthetic Concrete Pores

Synth-SHCC Synthetic Strain-Hardening Cement-Based Composite

T1 short Repetition Time and Time to Echo MRI scan (1 tissue type is bright: fat)

T1Gd Gadolinium T1-weighted

T2-FLAIR T2-Fluid Attenuated Inversion Recover

T2-weighted long Repetition Time and Time to Echo MRI scan (2 tissue types are bright: fat and water)

TN True Negatives

TP True Positives

UAV Unoccupied Aerial Vehicle

ViT Vision Transformer

VRAM Video Random Access Memory

XFEM Extended Finite Element Method