DISSERTATION

SUBMITTED

TO THE

COMBINED FACULTY FOR THE NATURAL SCIENCES AND MATHEMATICS

HEIDELBERG UNIVERSITY, GERMANY

FOR THE DEGREE OF

DOCTOR OF NATURAL SCIENCES

Put forward by

Master of Science Tomasz Kazimierz Konopczyński

Born in Zielona Góra, Poland

Oral examination:

Deep Learning Segmentation Algorithms for X-ray CT data

Advisors

Prof. Dr. Jürgen Hesser

Priv.-Doz. Dr. Christoph S. Garbe

Abstract

The segmentation task for 3D objects from X-ray CT volumetric data is of great significance for both industrial and medical applications. Deep learning techniques are narrowing the gap between human and machine capabilities in image segmentation. In this thesis we develop and discuss machine and deep learning techniques for semantic and instance segmentation. The techniques are evaluated on a dataset of CT scans of short glass fiber reinforced polymers prepared in cooperation with the University of Padova and on publicly available medical CT scans of lungs and liver. In addition to that, the last chapter is evaluated on a public and popular large-scale object detection, segmentation, and captioning dataset for a better comparison with the state-of-the-art.

The chapters are structured in the following way: In chapter 2 we explain the short glass fiber reinforced polymer data acquisition together with the reference setup for quantitative comparison of segmentation techniques. The data creation process involves parts manufacturing, CT scanning, CT simulation, computational model design, volume reconstruction and ground-truth preparation. The reference setup consist of metrics for instance and semantic segmentation tasks as well as of a baseline, Frangi vesselness method. In chapter 3 we present a first deep learning model for semantic segmentation of fibers from CT scans. The model outperforms all the other methods including feature-engineered and machine learning models. In chapter 4 we present a first deep learning model for instance segmentation of fibers from CT scans. The model outperforms the state-of-the-art by a significant margin and is arguably the first method which allows calculation of important fiber statistics based on single-fiber segmentation. The model consist of a fully convolutional branch for semantic segmentation, and an enhanced branch for instance segmentation via proposed embedding learning loss function. In chapter 5 we present our work on use of machine learning techniques for medical CT analysis. We use a *dictionary learning* model and extend it to a 3D for bronchial vessels segmentation from thorax CT scans. Then, we discuss and develop a *fully convolutional deep learning* model for the task of liver and liver lesion segmentation from liver CT scans. Lastly, we present the *Mask Mining* training approach for boosting the semantic segmentation machine learning models. In chapter 6 we present the idea of *the Plugin Networks* as a solution for inference under partial evidence. The proposed framework can generalize to a number of machine learning tasks and is evaluated on the task of hierarchical scene categorization, multi-label image annotation and scene semantic segmentation achieving state-of-the-art on each.

Zusammenfassung

Die Segmentierung von 3D-Objekten aus Röntgen-CT-Volumendaten ist sowohl für industrielle als auch für medizinische Anwendungen von großer Bedeutung. Mit Hilfe neuer Algorithmen aus dem Gebiet des Deep Learning (*auf deutsch: Tiefes Lernen*), einer Weiterentwicklung des „klassischen“ maschinellen Lernens, wird die Kluft zwischen den menschlichen und maschinellen Fähigkeiten bei der Bildsegmentierung zunehmend verringert. In der vorliegenden Arbeit werden Verfahren des „klassischen “ maschinellen Lernens sowie des Deep Learnings für die semantische und instanzielle Segmentierung entwickelt und diskutiert. Die Verfahren werden anhand eines Datensatzes von CT-Aufnahmen von kurzen glasfaserverstärkten Polymeren, die in Zusammenarbeit mit der Universität Padua erstellt wurden, und anhand frei verfügbarer medizinischer CT-Aufnahmen von Lunge und Leber bewertet.

Die Kapitel sind folgendermaßen aufgebaut: Im zweiten Kapitel erklären wir die Datenakquisition von glasfaserverstärkten Polymeren zusammen mit dem Referenzaufbau für den quantitativen Vergleich von Segmentierungstechniken. Der Datenerstellungsprozess umfasste die Fertigung der Teile, das CT-Scanning, die CT-Simulation, das Computermodelldesign, die Volumenrekonstruktion und die Vorbereitung der Ground-Truth-Daten. Der Referenzaufbau besteht aus Metriken für die Aufgaben der instanziellen und semantischen Segmentierung sowie der Frangi-Vesselness-Methode, welche als Basis dient. Im dritten Kapitel präsentieren wir ein erstes Deep-Learning-Modell für die semantische Segmentierung von Fasern aus CT-Scans. Dieser Algorithmus übertrifft alle anderen nicht Deep Learning basierten Methoden, welche auf Verfahren des maschinellen Lernens oder der nicht lernbasierten Merkmalsextraktion beruhen. Im vierten Kapitel präsentieren wir ein erstes Deep-Learning-Modell für die instanzielle Segmentierung der Fasern aus den CT-Scans. Das Modell übertrifft den aktuellen Stand der Technik deutlich und ist wahrscheinlich die erste Methode, die die Berechnung wichtiger Faserstatistiken auf der Grundlage einer Einzelfasersegmentierung ermöglicht. Im fünften Kapitel stellen wir die Verwendung unserer Techniken des maschinellen Lernens für die medizinische CT-Analyse vor. Wir verwenden einen *Dictionary - Learning* - Ansatz und erweitern es auf ein 3D-Modell für die Segmentierung von Bronchialgefäßen aus Thorax-CT-Scans. Anschließend diskutieren und entwickeln wir ein *Fully-Convolutional Deep Learning* - Modell für die Segmentierung der Leber und von Leberläsionen aus Leber-CT-Scans. Abschließend präsentieren wir den Trainingsansatz des *Mask Mining* zur Verbesserung der maschinellen Lernmodelle für die sematische Segmentierung. Im sechsten Kapitel präsentieren wir die Idee der *Plugin Networks* als eine Lösung für die Inferenz bei partieller Information. Das vorgeschlagene Framework kann auf eine Reihe von Aufgaben des maschinellen Lernens verallgemeinert werden und wird in Bezug auf die Kategorisierung hierarchischer Szenen, der Bildannotationen mit mehreren Labeln und die semantische Szenensegmentierung bewertet, wobei jeweils der aktuelle Stand der Technik übertroffen wird.

Acknowledgements

Throughout the work of this dissertation I have received a great deal of support and assistance. The work has been carried out from November 2015 to January 2020 at the Faculty of Mathematics and Computer Science in the Interdisciplinary Research Center for Scientific Computing under the supervision of Prof. Dr. Jürgen Hesser and Priv.-Doz. Dr. Christoph S. Garbe. The first three years were spent jointly at the Department of Radiation Oncology at the University Medical Center Mannheim and at Volume Graphics, a leading developer of analysis and visualization software for industrial CT data. The fourh year was spent at Tooploox, a software development company, which provides state-of-the-art solutions in AI and Machine Learning. The joint collaboration was possible thanks to the Marie Curie ESR Project INTERAQCT - International Network for the Training of Early stage Researchers on Advanced Quality control by Computed Tomography Funded by the European Commission's 7th Framework Programme FP7-PEOPLE.

First, I like to thank Prof. Dr. Jürgen Hesser for his ongoing support and guidance through each stage of the process during the last years while working on this thesis. For this, I am extremely grateful. I express my gratitude to Priv.-Doz. Dr. Christoph S. Garbe for his help and supervison of this thesis. In particular I like to thank Dr. Thorben Kröger for many hours of work he has invested in supervision and discussion of many aspects of my research during my stay at Volume Graphics. His help was instrumental in defining the path of my research. I thankfully acknowledge the contributions of Lei Zheng for his supervision during my stay at the Department of Radiation Oncology at the University Medical Center Mannheim. I like to thank Dr. Jitendra Rathore and Prof. Simone Carmignato for their collaboration and hosting during my secondment at the Unviersity of Padua. This thesis would not exist without their help. I gratefully acknowledge the contributions of Dr. Michał Koperski, Dr. Rafał Nowak, Piotr Semberecki and Dr. Tomasz Trzciński for the fruitful collaboration and work during the last year of my PhD studies at Tooploox. It is a pleasure to thank Karsten Roth and Danish Rathore for their outstanding work as students under my supervision. I am thankful to Susanne Maur, Ron Heiman, Nikolas Löw, Dr. Manuel Blessing and Dzmitry Stsepankou for the good time at the Experimental Radiation Oncology group and the reading club. I like to thank research fellows from the INTERAQCT project Andrea Buratti, Fabrício Borges de Oliveira, Aleksandr Amirkhanov, Massimiliano Ferrucci, Gabriel M Probst, Emmanuel Manu Dabankah, Alessandro Stolfi, Michele Pavan, Petr Hermanek, Evelina Ametova, Cao Wenchao, Nicholas Lippiatt, and Guruprasad Rao for the great time together. I acknowledge the supervisory board of the INTERAQCT project Prof. Dr. Ir. Wim Dewulf, Ir. ing. Kim Kiekens, Prof. Dr. Robert Schmitt, Prof. Leonardo De Chiffre, Dr. Stefan Kasperl, Prof. Dr. Johann Kastner, Dr. Ben Price, Dr. Markus Bartscher, Dr. Thomas Günther, Dr. ir. Tom Craeghs and Dr. Michael McCarthy. Finally, I like to thank my family for supporting me while working on this thesis. Most of all I like to thank my beloved wife Aleksandra for her unconditional love and understanding.

This thesis is dedicated to you.

Contents

Chapter 1

Introduction

1.1 Image Segmentation

Image segmentation is one of the most important objectives in image analysis and computer vision. It is the process of partitioning an input image into multiple pixel (or voxel) segments in order to assigning a label to every pixel in an image such that labeled pixels with the same labels shares certain desired characteristics. We distinguish two types of image segmentation: instance and semantic segmentation tasks. Semantic segmentation assigns same label for multiple objects of the same class. Contrarily, instance segmentation assigns multiple labels of the same class to objects within this class in an image. The segmentation task can be done on an arbitrarily dimensional objects, not only 2D images. In this thesis we focus on semantic and instance segmentation objects from 3D volumes.

1.2 X-ray Computed Tomography

Computed tomography (CT) is defined as "an imaging method in which the object is irradiated with X-rays or gamma rays and mathematical algorithms are used to create a cross-sectional image or a sequence of such images" [17]. In the process of tomography, X-rays are used to take a large number of finite 2D projections. These are then reconstructed using a tomographic reconstruction, multidimensional inverse problem algorithm to estimate a specific system from these projects, or to simply estimate a slice image of the scanned object [51]. These reconstructed slices are then stacked to from a 3D volumetric representation of the object of interest which can be then used in a wide range of applications. In this thesis we focus on segmentation of objects from 3D X-ray CT scans.

1.3 Fiber Segmentation from 3D X-ray CT

The most popular method to date for automatic fiber segmentation (or more generally, cigar-like structures) is the Frangi vesselness filter [36]. This cigar-like structure enhancement filter [131] derives structural information from the Hessian eigenvalues $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$. For computation of the Hessian, a number of Gaussian second derivatives at scales σ_s are used. Gaussian kernels allow separable convolution, which speeds up computation time. The Gaussian second order derivative of image I at scale σ and point x is given by

$$\frac{\partial^2 I_\sigma}{\partial x^2} = I(x)\frac{\partial^2 G(\sigma, x)}{\partial x^2} \tag{1.1}$$

The Frangi vesselness filter $V(x)$ uses a combination of three measures to distinguish structures in the image, and is given by

$$V(\sigma, x) = \begin{cases} 0, & \text{if } \lambda_2, \lambda_3 > 0 \\ F, & \text{otherwise} \end{cases} \tag{1.2}$$

$$F = \Big(1 - \exp\Big(-\frac{R_A^2}{2\alpha^2}\Big)\Big) \cdot \exp\Big(-\frac{R_B^2}{2\beta^2}\Big) \cdot \exp\Big(1 - \exp\Big(-\frac{S^2}{2c^2}\Big)\Big) \tag{1.3}$$

where the first measure $R_B = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}}$ quantifies deviation from a blob-like structure, the second $R_A = \frac{|\lambda_2|}{\lambda_3|}$ quantifies the difference between plate-like and cigar-like structures and the third $S = ||H||_F = \sqrt{\Sigma_i \lambda_i^2}$ quantifies the presence of the background noise. α, β, c are real valued positive parameters of the filter. The filter is finally embedded in a non-maximum suppression multi-scale framework

$$V(x) = \max_{\sigma} V(\sigma, x) \tag{1.4}$$

We use this method as a reference in the following chapters.

1.4 Dictionary Learning

Feature generation via Dictionary Learning is an unsupervised problem, where from a number of patches the algorithm learns a set of elements that allow for an optimal representation. Dictionary learning based algorithms were among the most popular feature learning techniques before deep learning.

Dictionary learning usually requires a set of Gaussian pyramids by convolving input volumes with Gaussian kernels and subsampling them [83]. In our dictionary learning setup, from the volume data we randomly select a batch of 3D patches $p^{(i)} \in \mathbb{R}^n$ (where n is the number of voxels of the patch) which we use as an input to the sparse coding algorithm for learning a dictionary $D \in \mathbb{R}^{n \times d}$ of d elements, where each column $D^{(j)}$ is one element. We train the dictionary by minimizing the LASSO problem with L1-penalization to ensure the sparsity of the vector $x^{(i)}$ regularized by the parameter λ. That is, we optimize

$$\min_{D, x^{(i)}} \sum_i ||Dx^{(i)} - p^{(i)}||_2^2 + \lambda ||x^{(i)}||_1 \tag{1.5}$$

$$\text{subject to } ||D^{(j)}||_2^2 = 1, \forall j$$

over the sparse codes $x^{(i)}$ and the dictionary, D. We use it for the task of bronchial vessels segmentation from thorax scans in chapter 5.

1.5 Deep Learning

Deep learning architectures are successfully applied to a range of image analysis and processing problems in natural sciences [76, 34, 35, 123]. Most importantly, from the perspective of this thesis, deep learning show great improvement over other methods in the task of semantic and instance segmentation for both natural 2D images and 3D CT volumes [118, 19]. Similar solutions are found for the problem of 2D instance segmentation. Faster R-CNN [106] and the Mask R-CNN [46] architectures are examples of region-proposal-based techniques which are the state-of-the-art for common scene-understanding datasets like COCO [79] or ImageNet [23]. However, it is not clear how this approach can be extended to 3D volumetric data with densely packed or tangled objects like fibers in fiber reinforced polymers or bronchial vessels in thorax scans. In this thesis we focus on deep learning segmentation techniques 3D X-ray CT scans of short glass fiber reinforced polymers (SFRP), lungs and liver.

1.5.1 Semantic Segmentation

The fully convolutional network (FCN) [118] is arguably the first deep learning model for the semantic segmentation task. The difference to a standard convolutional network (CNN) is that at the end of the network, fully connected layers are replaced with convolutional layers. In contrary to the classification task, when the desired output is just one class per image, for the semantic segmentation task we assign one class per pixel on an image. The loss function definition however can stay the same as for the classification task.

In the medical community, the most popular architecture for the semantic segmentation task is the the U-Net architecture [18] which is designed in an Autoencoder fashion [5]. Modern U-Net architectures with skip connections and auxiliary tasks allow the network to enhance both low and high frequency features of the object.

In this work we use two commonly used binary loss functions for training U-Nets and FCNs. Namely The cross entropy and the dice loss functions. The standard voxel-wise binary cross entropy loss is defined as:

$$\mathcal{L}_{CE} = \sum_i y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \tag{1.6}$$

where $y_i \in \{0, 1\}$ is the i-th the true binary label, and $\hat{y}_i \in \{0, 1\}$ is the i-th predicted label.

the dice loss is defined as:

$$\mathcal{L}_{Dice} = -\frac{2 \sum_i \hat{y}_i y_i}{\sum_i \hat{y}_i + \sum_i y_i} \tag{1.7}$$

The loss functions can be further modified explicitly by adding a weight map W_i for some regions (for instance borders of objects) to put more emphasis on them. It is also common to apply the weighted sum of such a pixel weighted cross-entropy and Dice as a loss function for the network output and the auxiliary outputs merging them together. It brings us to a more generalized form of a binary loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[\sum_k \left[(1 + w_{ik})^{\lambda} \cdot y_{ik} \cdot \log \hat{y}_{ik} \right] + \alpha \frac{2 \cdot |Y_i \odot \hat{Y}_i|}{|Y_i| + |\hat{Y}_i|} \right], \tag{1.8}$$

where $\hat{Y}_i$ and Y_i indicate the predicted and ground truth mask, respectively, on one image. w_{ik}, y_{ik} and $\hat{y}_{ik}$ denote k-th element of W_i, Y_i and $\hat{Y}_i$, respectively. λ is a weight map factor, α a scalar weight factor, N is the batch size and $\odot$ the Hadamard product.

We make use of them in chapters on semantic segmentation.

1.5.2 Instance Segmentation via Embedding Learning

There are numerous works in which authors try to come up with different ideas for instance segmentation in the deep learning framework with the Mask R-CNN [46] being the most successful and popular to date for 2D images. However, it is not clear how this approach can be extended to 3D volumetric data with densely packed objects like fibers in SFRP. This is why for our 3D problem, we have opted for alternative deep learning methods for instance segmentation and propose a novel 3D deep learning architecture based on embedding learning.

By extending to 3D loss functions introduced by [10] inspired by work of [129] we are able to learn embeddings which create clusters in a multi-dimensional space making it possible to distinguish fibers from each other. The loss consists of three terms: $\mathcal{L}_v$ keeps voxels belonging to the same object close to each other, $\mathcal{L}_d$ which forces a minimal distance between clusters of different objects, and $\mathcal{L}_r$ which regularizes the cluster centers to be close to the origin. The terms are defined as:

$$\mathcal{L}_v = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}[||\mu_c - x_i|| - \delta_v]_+^2 \tag{1.9}$$

$$\mathcal{L}_d = \frac{1}{C(C-1)}\sum_{c_A=1}^{C}\sum_{c_B=1, c_A\neq c_B}^{C}[\delta_d - ||\mu_{c_A} - \mu_{c_B}||]_+^2 \tag{1.10}$$

$$\mathcal{L}_r = \frac{1}{C}\sum_{c=1}^{C}||\mu_c|| \tag{1.11}$$

where C is the number of objects in the ground truth patch (clusters), N_c is the number of voxels that corresponds to the object c, x_i is the embedding in the final embedding layer, μ_c is the mean of the embedding of object c, $||\cdot||$ is the L_2 norm, and $[x]_+ = max(0, x)$. The parameters δ_v and δ_d are used to control the desired positions of the clusters. The final loss for the embedding learning L_{embd} is a sum of the previous components.

$$\mathcal{L}_{embd} = \alpha\mathcal{L}_v + \beta\mathcal{L}_d + \gamma\mathcal{L}_r \tag{1.12}$$

where α, β and γ control the strength of the corresponding term.

A detailed review of the Embedding Learning performance on the SFRP CT scans is given in the chapter 4.

1.6 Plugin Networks

The availability of partial information (*partial evidence*) about an image, made available at test time, can improve accuracy of a pre-trained networks [52, 128]. We propose a novel method to incorporate partial evidence in the inference of deep convolutional model. Contrary to the existing, top performing methods, which either iteratively modify the input of the network or exploit external label taxonomy to take the partial evidence into account, we add separate network modules ("Plugin Networks") to the intermediate layers of a pre-trained convolutional network.

Let's assume that we have a CNN model $F(\mathbf{x};\mathbf{w})$, where $\mathbf{x}$ is an input image and $\mathbf{w}$ are the parameters. The model F is already trained on some task (*e.g.* single or multi-label classification, scene segmentation). The parameters $\mathbf{w}$ were trained on input images X and input labels Y. Now let's assume that some labels $\bar{Y}$ are available and known at inference time. In the following definitions without loosing generality, we will assume that only one Plugin Network is attached to the base model. We define the Plugin Network model F_p with parameters $\mathbf{w}_p$ as

$$\mathbf{r} = F_p(\bar{\mathbf{y}};\mathbf{w}_p). \tag{1.13}$$

The model takes the partial evidence $\bar{\mathbf{y}} \in \bar{Y}$ as an input. The output $\mathbf{r}$ of the plugin can be attached to the output vector $\mathbf{z}$ of some layer of the base model F:

$$\tilde{\mathbf{z}} = \mathbf{z} \oplus \mathbf{r}, \tag{1.14}$$

where the sign $\oplus$ can have the following meaning:

- additive: $\tilde{\mathbf{z}} = \mathbf{z} + \mathbf{r}$,
- multiplicative: $\tilde{\mathbf{z}} = \mathbf{z} * \mathbf{r}$,
- residual: $\tilde{\mathbf{z}} = \mathbf{z} + \mathbf{z} * \mathbf{r}$.

In this way the Plugin Network F_p adapts the output vector $\mathbf{z}$ of the base model F under presence of available partial evidence. The eq. (1.14) defines how the Plugin Network F_p is attached to the base network F, thus a joint model can be defined as:

$$\tilde{F}(\mathbf{x},\bar{\mathbf{y}};\mathbf{w},\mathbf{w}_{p_i}). \tag{1.15}$$

In general, several Plugin Networks can be attached simultaneously to a number of layers of the base model F.

A detailed review of the Plugin Network performance is given in the chapter 6.

Chapter 2

Reference Setup for Quantitative Comparison of Segmentation Techniques

2.1 Outline

In this chapter we provide a dataset of CT scans of short fiber reinforced polymer (SFRP) together with a groundtruth and an evaluation criteria for automatic segmentation methods. The dataset containing the experimental scans, the synthetic scans and corresponding ground truths together with the algorithms is available at `http://ipm-datasets.iwr.uni-heidelberg.de`. We discuss the dataset preparation, together with evaluation metrics which we propose to use as a reference for work on semantic segmentation of SFRP for learnable methods as well as an investigation of CT scans of SFRP composite materials. Parts of this chapter have been published in the International Conference on Industrial Computed Tomography (iCT) 2017 proceedings [69] and in the Journal of Nondestructive Evaluation, Diagnostics and Prognostics of Engineering Systems [104].

2.2 Introduction

Quantitative assessment of fiber characteristics in composite parts is of great significance for material science, device design, and production in order correlate them with the fiber induced mechanical properties. X-ray computed tomography (CT) is being successfully used as a three-dimensional non-destructive measuring technique for the analysis of these fiber characteristics (mainly the fiber orientation and fiber volume content) in fiber-reinforced composite materials.

2.2.1 Short fiber reinforced polymer composites

Short fiber reinforced polymer composites constitute of relatively short and variously aligned fibers distributed in a continuous polymer matrix [38]. Glass, carbon, graphite, and Kevlar are commonly used materials for these fibers. In general, SFRPs are characterized by their versatile properties such as low manufacturing cost, which render them extensively useful in fields like automotive, electronics, marine, aerospace, and household applications. Superior mechanical properties as compared to polymers are the main advantages of SFRPs. However, these properties are greatly influenced by the fiber characteristics such as fiber-matrix interface strength, fiber volume fraction, fiber orientation distribution, and fiber length distribution [38, 39, 127]. Moreover, injection molding (IM) is one of the most commonly employed manufacturing processes for large-scale production of SFRP composites. Precision manufacturing of parts produced by IM recently attracted large attention for electronics applications, including connectors, because of their increasing market trends [127]. Despite the trend of miniaturization observed for IM applications, connectors remain relatively large, because of their complex design, which poses several manufacturing issues [88]. In particular, the thin-wall that characterizes their typical geometry constitutes a major manufacturing constraint [80]. Hence, the commercial breakthrough of new and smaller connectors strongly depends on the necessity to develop low-cost mass production technologies, which can provide dimensional accuracy and good part quality [116].

It has been demonstrated in literature that quality and dimensional accuracy of injection-molded composite parts mainly depend on the injected polymer [93], part geometry, mold design, selection of process variables [120], and fiber orientation [85]. In order to achieve the desirable fiber material characteristics in the final molded product, the manufacturing process has to be optimized by establishing a correlation of the processing parameters with the fiber characteristics [103, 86]. For example, Oumer et al. [97] presented a review of the effect of processing parameters on fiber orientation and the effect of mold temperature on motion behaviour of short glass fibers was studied by Li et al. [78]. Nevertheless, these studies very much rely on the accuracy of the measurement technique for fiber analysis. A comparison between optical and tomographic methods for fiber analysis was performed by Bernasconi et al. [7]. They pointed out that the optical method requires a simpler experimental setup but it is mostly destructive. On the other hand, the tomographic technique is non-destructive with exception in case of ROI (region of interest) scanning but requires expensive experimental facilities. In the last decade, X-ray computed tomography (CT) has evolved as a powerful technique for industrial applications owing to the continuous improvement towards the increase of accuracy and traceability [50, 16].

In the field of composite materials, CT is being used increasingly for various fiber-based analyses [121, 119]. For instance, CT for analysing fiber orientation [130] is very efficient and advantageous, since characterization of fiber orientation and of the skin-core morphology by optical observations of cross sections of the moldings can be very complex for thin-wall parts. In addition, as a non-destructive technique, CT eliminates distortions introduced by sample cutting and preparation [119]. However, the accuracy of CT based analyses is sensitive to the selection of various scanning parameters (current, voltage, and exposure time), spatial resolution (voxel size) and the system limitations (source, detector properties) [50]. In case of SFRP composites, the spatial resolution becomes very crucial due to the small size (diameter) of fibers. Kastner et al. [63] reported the use of high-resolution X-ray CT for fiber reinforced polymers. If high-resolution is demanded, only a region of interest (ROI) should be considered for achieving good spatial resolution instead of using the entire part [94]. The aim of this work is to investigate the dependence of the CT based fiber characterization on spatial resolution and scanning parameters by using a synthetic fiber volume with known ground truth as it helps in understanding the deviations from the true values. In comparison, a real injection molded thin-walled part is also examined with similar CT settings, to predict the best case scenarios for analysing real injection molded products of the same materials e.g. micro connectors.

2.2.2 Need for Quantitative Comparison of Segmentation Techniques

Following the investigation of model parameters, we continue to discuss an approach to automatically segment individual fibers. Comparing different algorithms for segmenting glass fibers in industrial computed tomography (CT) scans is difficult due to the absence of a standard reference dataset. We introduce a set of annotated scans of short-fiber reinforced polymers (SFRP) as well as synthetically created CT volume data together with the evaluation metrics. We suggest both the metrics and this data set as a reference for studying the performance of different algorithms. The real scans are acquired by a Nikon MCT225 X-ray CT system. The simulated scans are created by the use of an in-house computational model and third-party commercial software. For both types of data, corresponding ground truth annotations are prepared, including hand annotations for the real scans and STL models for the synthetic scans. Additionally, a Hessian-based Frangi vesselness filter for fiber segmentation is implemented and open-sourced to serve as a reference for comparisons.

The influence of fiber characteristics (e.g. fiber orientation, fiber length distribution and the percent composition in the final product) on the mechanical properties of SFRP composites is of particular interest and significance for manufacturers [37]. These fiber characteristics are affected by the processing conditions (melt temperature, mold temperature, packing pressure and cooling time); therefore, reliable information about fiber characteristics is much needed for the process optimization during the product development phase. In this chapter we focus on the (single)-fiber segmentation of volumetric images which is needed for most of the precise local measurements of SFRP composite characteristics. A binary segmentation of fibers/non-fibers leads to a direct measurement of the fiber ratio in a volume. Moreover, a single-fiber segmented volume can be used to retrieve the correct length and orientation distribution in a volume.

A number of methods used for fiber segmentation are described in literature [31, 101, 135]. However, it is difficult to compare their performance. In order to judge the suitability of a given algorithm for the task of fiber segmentation, it is necessary to have reliable measures for the quality and accuracy of different methods. To address this problem we propose a reference dataset on which a quantitative comparison of segmentation techniques can be performed. Following the "Grand Challenge" [22] concept used in the medical imaging community, we host the data in a similar framework. We provide a publicly available standard dataset of SFRP composite CT scans together with its ground truth and a set of metrics, with which different fiber segmentation techniques may be verified. We use X-ray scans of micro-injection molded parts from a commercially used SFRP material for a case study. To support the hand-annotations, we develop a post-processing method for ground truth preparation. Even with the use of our post-processing method, hand-annotating the data is very time-consuming and results in only a low number of annotated fibers. Therefore, we additionally create a computational model of SFRP composites. By using the model, we perform a number of synthetic CT scans, which are provided in the dataset. The synthetic scans are similar to the real data, and thus the same set of algorithms might be verified on them. For the evaluation criteria we use the Dice coefficient [43] and Adjusted Rand Index [55], which are standard metrics commonly used to validate the quality of segmentation techniques.

2.2.3 Related Work

There are many reference datasets available to compare different methods for various task specific applications. The framework of the "Grand Challenge" organized by the Consortium of Open Medical Image Computing is very popular in the field of image processing. At the moment of writing this publication it is currently hosting 134 individual challenges [22] (i.e. annotated datasets together with evaluation metrics and results of different algorithms). One example of a successful and similar challenge to ours is the "VESsel SEgmentation in the Lung 2012 challenge" [114] in which the participants were asked to segment vessels from CT thorax scans. Due to the similar tubular-looking structure of vessels, the algorithms for fiber segmentation are very similar to those for vessel segmentation. Acquiring scans in high resolution is time consuming and costly. Therefore, in this work we consider only scans acquired by a CT system with low (3.9 µm and 8.3 µm) resolution. The described baseline method has been primarily designed for vessel segmentation. Thus the name Frangi vesselness filter.

Creating a model of a specimen is a common method of evaluating an algorithm. A similar approach has been proposed to evaluate 3D fiber orientation algorithms based on X-ray computed tomography models of SFRP [108]. This work however, omits the X-ray imaging simulations. Similarly, Z. Bliznakova et al. proposed a computational model for carbon fiber-reinforced polymers [9]. The model was later used in an entire pipeline for carbon fiber-reinforced polymers modeling for X-ray imaging simulation together with a CT scan simulation [122]. Unfortunately, to the best knowledge of the authors none of the models is publicly available. In our work, we first create a geometrical model of SFRP composite, then simulate an X-ray imaging and perform CT reconstruction of the simulated projections. Our simulation is set up to match the parameters of the material used for the experimental scans of the SFRP specimen on a Nikon MCT225 X-ray CT system.

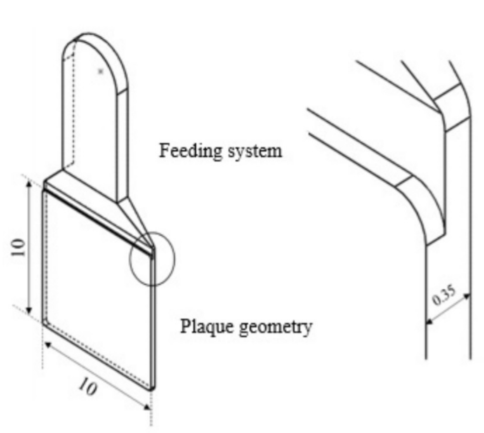
2.3 Material and Methods

2.3.1 Dataset preparation

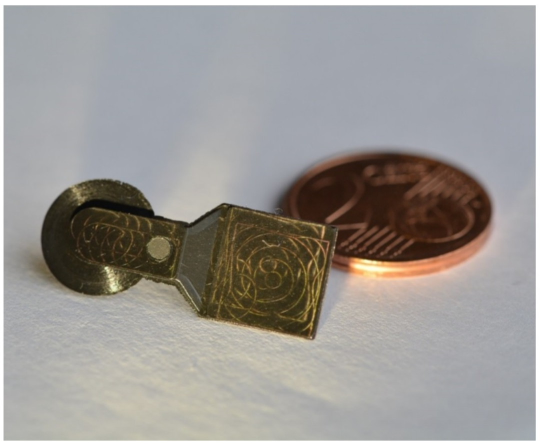
This section consists of the information about acquiring the two types of fiber composite parts used for this work: a synthetic volume (computationally generated STL based model) and a real part manufactured from micro injection molding process.

2.3.2 Manufacturing a composite part

The part we use for our dataset was manufactured by micro injection molding using PBT-10% GF, a commercial polybutylene terephthalate PBT (BASF, Ultradur B4300 G2) reinforced with short glass fibers (10% in weight). The main application of the material is a micro connector which consists of multiple thin-walled features and demands for good dimensional accuracy. In order to study the effect of fiber orientation on shrinkage behavior of the part, we choose a simplified thin-walled plaque geometry of $10 \times 10 \times 0.35\,\text{mm}^3$ (Fig. 2.1), which is



(a)



(b)

FIGURE 2.1: Part used for this study. (a) Plaque geometry of 10 mm × 10 mm × 0.35 mm (the circle indicate the enlarged area on the right side showing the thickness) and (b) the real part.

used to evaluate shrinkage and the orientation tensor using µ-CT according to the conventional injection molding standards and methodologies proposed in the literature for thin-wall parts [13, 3]. We choose The extremely small thickness of 0.35 mm to understand the fiber orientation trends in thin-walled parts in comparison to relatively higher thicknesses as reported in [86].

We use the state-of-the-art micro injection molding machine (Wittmann Battenfeld, MicroPower 15) for the manufacturing of the plaque geometry (10 mm × 10 mm × 0.35 mm). The machine is characterized by a separated 14 mm plasticizing screw and a 5 mm injection plunger. A mold heating system is implemented by using four electrical cartridges, two for each mold half, and two thermocouples, in order to guarantee a stable mold temperature. A maximum injection speed of 750 mm/s and a maximum clamping force of 150 kN is used for the experiments.

2.3.3 Modelling a composite part

A synthetic fiber volume of known fiber distribution and orientation is generated with using characteristics of a real material, which is also used in manufacturing a real part by means of injection molding. The material selected for this study is a commercial grade polybutylene terephthalate (PBT) compound (BASF, Ultradur B4300 G2) for microelectronics applications (such as housings, plugs and micro connectors). It is reinforced with 10 % (in weight) short glass fibers. PBT is a semi crystalline thermoplastic polymer that is characterized by high shrinkage, making its processing critical in terms of dimensional accuracy. The introduction of fibers further complicates the prediction of the material behaviour and thus arises the need to understand the behaviour to achieve a desired dimensional accuracy. In order to generate the virtual fiber model, the main a priori information is the fiber volume fraction (FVF), which can be deduced from the equation (1) using the provided fiber weight fraction (FWF) from the material specifications [98].

$$FVF = \left[1 + \frac{\rho_f}{\rho_m}\left(\frac{1}{FWF} - 1\right)\right]^{-1} \tag{2.1}$$

TABLE 2.1: Fiber properties used for modeling the synthetic volume used in this study.

Property	**Value**
Fiber length	500 ± 100 [μm]
Fiber radius	6.5 [μm]
Density of glass fibers (S-glass)	2.54 [g/cm^3]
Density of matrix (PBT)	1.31 [g/cm^3]
Specimen dimensions	$2 \times 2 \times 2mm^3$

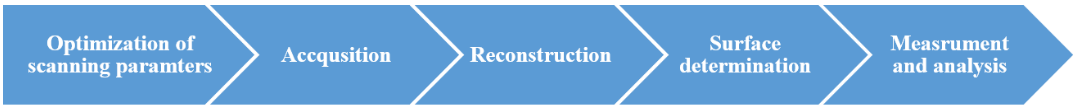


FIGURE 2.2: X-ray CT measurement work flow.

where, ρ_m and ρ_f are the density of matrix and fibers respectively (in g/cm^3). The other fiber properties required for the modeling are provided in Table 2.1.

It should be noted that, the purpose of using the material characteristics for the synthetic volume is to have a comparable fiber volume content. However, the fiber length distribution and orientation are not indented to be identical with the real injection molded part as it does not affect the current investigation. The fiber length is sampled from a normal distribution with a mean and standard deviation of 500 µm and 100 µm respectively. On the other hand, the diameter is chosen as 13 µm, which is measured from a high resolution CT scan of a micro pellet of PBT. The fibers are assumed as solid cylinders of fixed density (glass) with given diameter and varied length. The surrounding matrix is created using a volume of (2 $\times$ 2 $\times$ 2) mm^3 to match the fiber calculated fiber volume of 5.40 % $\pm$ 0.05 %.

2.3.4 X-ray CT: simulations and experiments

X-ray CT based characterization is an extensive task where a number of steps are involved as mentioned in the Fig. 2.2. The final results depend on the selection of the scanning parameters which may need to be optimized considering the size and the material of the part. The projections are then acquired with the optimized parameters and reconstructed into a 3D volumetric dataset and a surface is defined. In this work, projections are acquired in the form of simulation of the scanning procedure for the synthetic fiber model and scanning the manufactured part physically.

A metrological micro CT system (Nikon Metrology, X-Tek MCT 225) is utilized for acquiring the projections for the manufactured composite part. The maximum permissible error (MPE) for this system is given by: (9 + L/50) µm (where L is the measured length expressed in millimeters) [14]. The metrological performances of the CT system is evaluated using specific procedures and a fiber-based calibrated object [15, 84]. On the other hand, aRTist 2 (Analytical RT Inspection Simulation Tool) software package from BAM, Germany, is used for simulation of acquisition procedure [60]. The generated STL model of fibers is uploaded and embedded inside a cube geometry, which serves as the matrix. The material density and

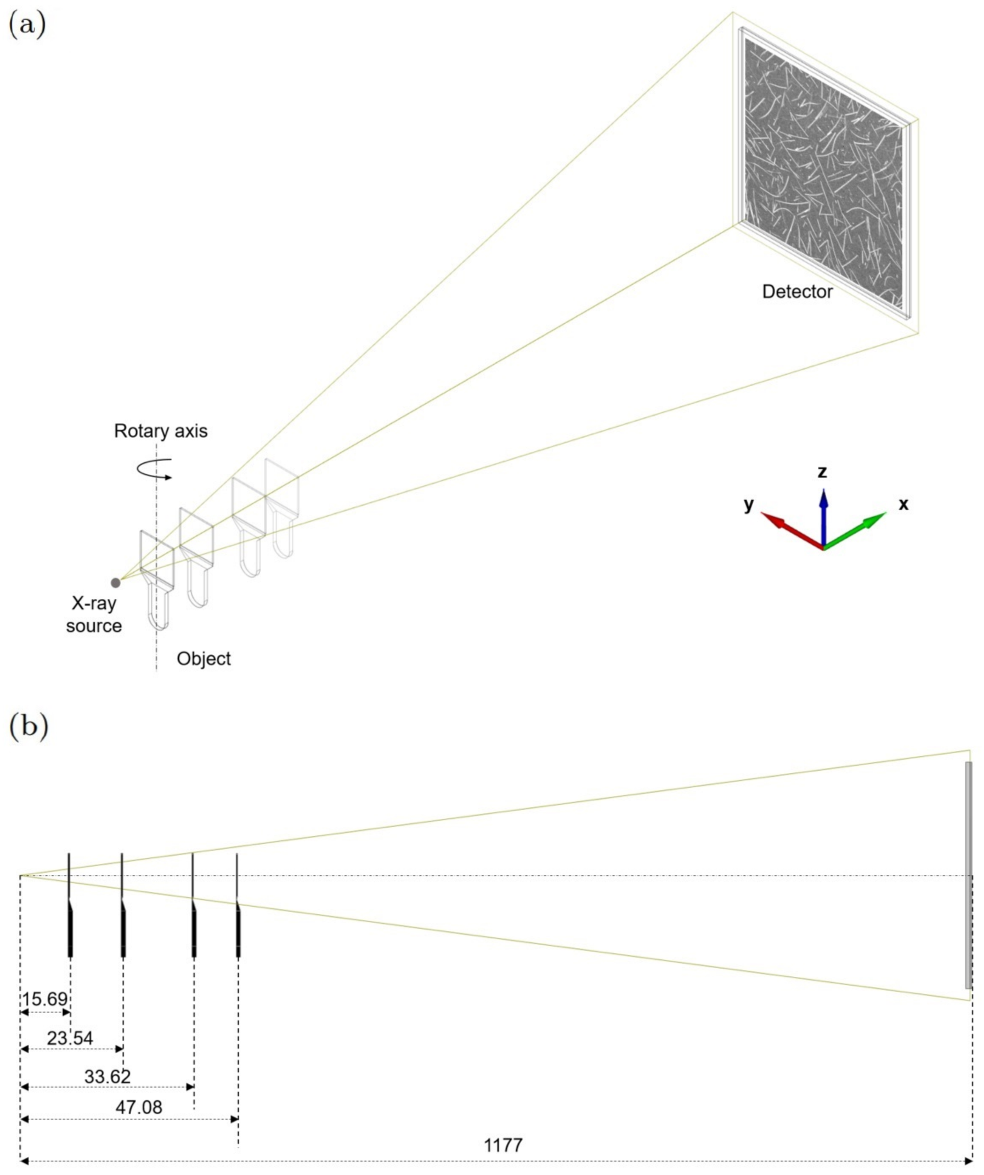


FIGURE 2.3: Schematic representation of the CT scanning layout for all the magnifications: 3D view (a) and 2D view (b) (dimensions are in mm).

composition are required for the X-ray attenuation, which are used as reported in Table 2.1. The detector and source parameters are set based on the characteristics of the Nikon MCT225 X-ray CT system. The detector size is set to 2048 $\times$ 2048 pixels with a pixel resolution of 0.2 mm. For the noise factor, the signal-to-noise ratio (SNR) of the air of a real experimental projection is checked and adapted to the simulation.

The part placement for the scanning is schematically shown in Fig. 2.3. The source-to-detector distance (SDD) is 1,177 *mm* and four different source-to-object distances (SOD) are chosen, resulting in different magnifications (see Table 2.2) with corresponding resolution defined by a cubic voxel (volumetric pixel). The four resolutions chosen during this study are referred as best resolution (BR), high resolution (HR), medium resolution (MR) and low resolution (LR) corresponding to the voxel sizes of 2.7 µm, 4 µm, 5.7 µm, and 8 µm respectively. Higher resolution is achieved by placing the object closer to the source but at the cost of smaller field of view (FOV). Only the low and medium resolutions (i.e. LR and MR) are

TABLE 2.2: Different magnification settings.

SOD [mm]	**Voxel size** [μm]	**Notation**
15.69	2.7	BR
23.54	4	HR
33.62	5.7	MR
47.08	8	LR

TABLE 2.3: CT data acquisition parameters.

Factor	**Set-1**	**Set-2**
Voltage [kV]	120	95
Current [μm]	71	74
Exposure time [s]	1.4	2.8
Number of Projections	2,000	1,800
Averaging (frames per projection)	4	2
Scanning time [$min.$]	$\sim$190	$\sim$150

resulting into the full scan of the plaque geometry and the other two are partial scan which is also termed as ROI scan as depicted in Fig. 2.3 b. In order to have a comparative study of all the four resolutions a common ROI is considered as explained in Fig. 2.8. The X-ray projections are acquired for a complete rotation cycle and subsequently reconstructed into a 3D volumetric data set. The CT scanning parameters are chosen with consideration of material density and by analysing the grey value histogram on the projections. As mentioned in Table 2.3, two sets of CT parameters are finalized to understand their effect on the results.

2.3.5 Generated fiber model

The proposed computational model of SFRP composite is set to match the fiber content of 5.40 %, which is calculated using the densities of glass fiber and PBT matrix. The orientation vector (P) of a fiber is represented by polar (θ) and azimuthal angle (ϕ) in a spherical coordinate system as described in Fig. 2.4 a; and the resultant ϕ and θ for the obtained fiber model are shown in in Fig. 2.4 b and c.

2.3.6 Reference algorithm

As baseline reference, we propose to use a tubular structure enhancement filter [131]. Concretely we employ a Frangi vesselness filter [36], which derives structural information from the Hessian eigenvalues $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$. For computation of the Hessian, a number of Gaussian second derivatives at scales σ_s are used. Gaussian kernels allow separable convolution, which speeds up computation time. The Gaussian second order derivative of image

TABLE 2.4: Synthetic fiber model statistics used in this study.

Attribute	Explanation	Value
Number of fibers	n	6,628
Total fiber volume [mm^3]	$n\pi r^2 l_{avg}$	0.432
Average length [μm]	l_{avg}	491.231
Fiber volume fraction [%]	-	5.399

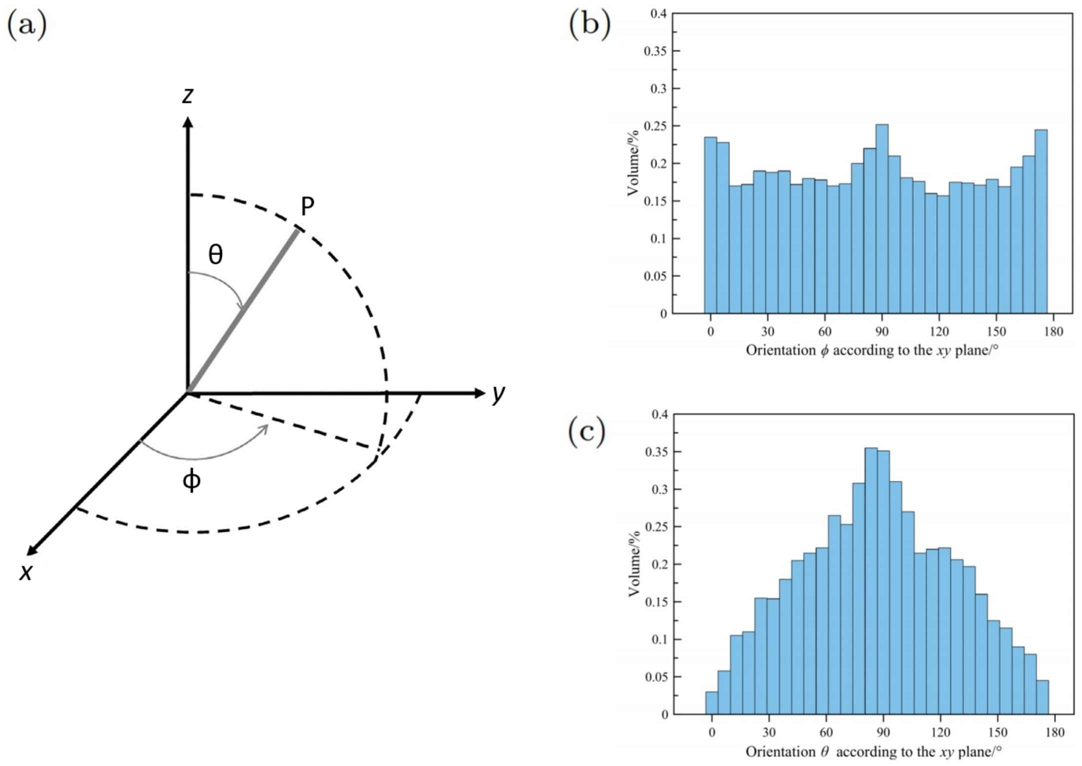


FIGURE 2.4: Description of fiber orientation under a polar coordinate system. (a) the polar coordinates, (b) the histogram of orientation for the azimuthal angle ϕ (b) and the histogram of orientation for the polar angle θ (c) of the the obtained computational fiber model.

I at scale σ and point x is given by

$$\frac{\partial^2 I_\sigma}{\partial x^2} = I(x)\frac{\partial^2 G(\sigma, x)}{\partial x^2} \tag{2.2}$$

The Frangi vesselness filter [36] $V(x)$ uses a combination of three measures to distinguish structures in the image, and is given by

$$V(\sigma, x) = \begin{cases} 0, & \text{if } \lambda_2, \lambda_3 > 0 \\ F, & \text{otherwise} \end{cases} \tag{2.3}$$

$$F = \left(1 - \exp\left(-\frac{R_A^2}{2\alpha^2}\right)\right) \cdot \exp\left(-\frac{R_B^2}{2\beta^2}\right) \cdot \exp\left(1 - \exp\left(-\frac{S^2}{2c^2}\right)\right) \tag{2.4}$$

where the first measure $R_B = \frac{|\lambda_1|}{\sqrt{|\lambda_2\lambda_3|}}$ quantifies deviation from a blob-like structure, the second $R_A = \frac{|\lambda_2|}{\lambda_3|}$ quantifies the difference between plate-like and cigar-like structures and the third $S = ||H||_F = \sqrt{\Sigma_i \lambda_i^2}$ quantifies the presence of the background noise. α, β, c are real valued positive parameters of the filter. The filter is finally embedded in a non-maximum suppression multi-scale framework

$$V(x) = \max_{\sigma} V(\sigma, x) \tag{2.5}$$

As baseline reference, we implement the 3D version of the Frangi vesselness filter. Additionally, for the local orientation distribution, a structure tensor based algorithm from the 2D implementation of ImageJ plugin OrientationJ [102] was implemented and extended to 3D. It serves as a support tool to evaluate additional statistics of the datasets.

2.3.7 Evaluation criteria

For evaluation criteria, we propose the two following common metrics [43, 55]. One for the binary "fiber/non-fiber" segmentation task and one for the single-fiber segmentation task.

For the binary classification, we use the mean Dice Coefficient [43]. It compares the predicted segmentation on a pixel level with the ground truth (labeled volumes). Defining the binary ground truth labels as a cluster B and the corresponding predicted binary labels as a cluster B', the Dice coefficient index D is defined by

$$D(B, B') = \frac{2 \times |B \cap B'|}{|B| + |B'|} = \frac{2 \times TP}{2 \times TP + FN + FP} \tag{2.6}$$

where the intersection operation is the voxel-wise minimum operation, and $|\cdot|$ is the integration of the voxel values over the complete image, TP is the true positive, FP false positive and FN false negative. The score varies between 0 and 1, where 1 means a perfect match between the algorithm output and the ground truth mask, and 0 complete mismatch.

For the instance (single-fiber) segmentation, we use the Adjusted Rand Index [55]. We find it more informative in the context of SFRP data over the more common mean average precision metric. Defining the ground truth labels as a cluster $C = \{C_1, \ldots, C_k\}$ and the corresponding predicted labels as a cluster $C' = \{C'_1, \ldots, C'_l\}$, the Adjusted Rand Index R_a is of a form:

$$R_a(C, C') = \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} \binom{m_{ij}}{2} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3} \tag{2.7}$$

where $m_{ij} = |C_i \cap C'_j|$, $t_1 = \sum_{i=1}^{k} \binom{|C_i|}{2}$, $t_2 = \sum_{j=1}^{l} \binom{|C'_j|}{2}$, $t_3 = \frac{2t_1t_2}{n(n-1)}$ and n is the number of voxels in the volume. The Rand Index varies from 0 to 1, where 1 means a perfect match between the algorithm output and the ground truth mask.

For the baseline method, we implement and open source the 3D Frangi veselness filter [36]. For instance, our 3D Frangi vesselnes filter implementation achieves a score of 0.704 for the task of binary segmentation on one of the synthetic volumes.

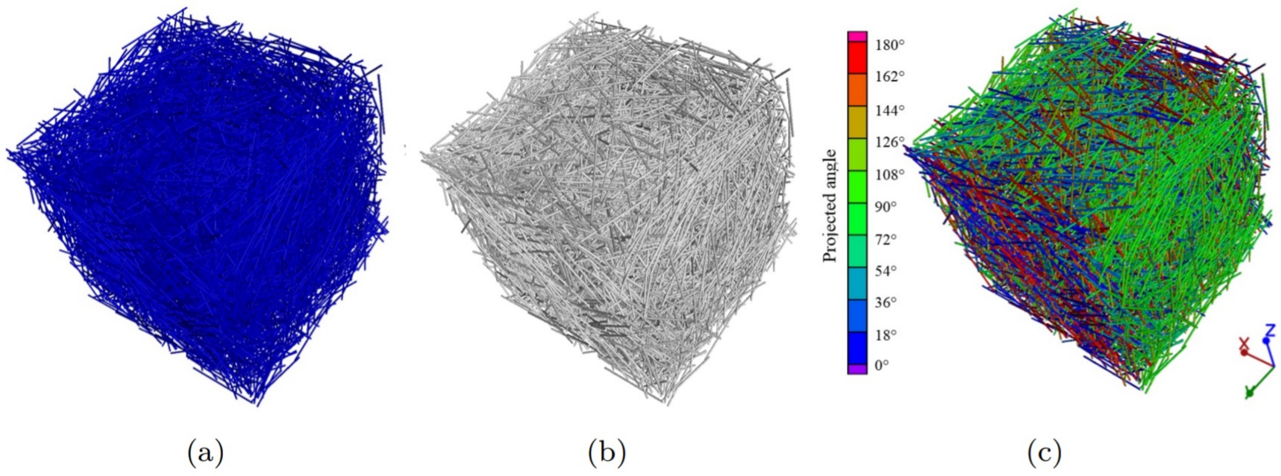


FIGURE 2.5: Work-flow from fiber modeling to CT fiber analysis. (a) an STL model from the computational model, (b) segmented fibers from a simulated CT scan, (c) orientation analysis on the simulated part.

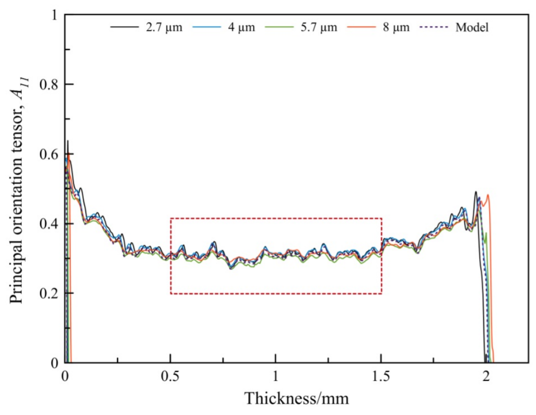
2.4 Results

2.4.1 Simulation results

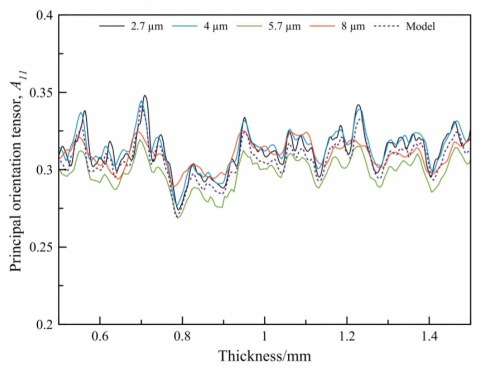
The reconstruction of the CT projections is performed by use of the standard filtered back projection (FBP) reconstruction algorithm implemented in the reconstruction module of the commercial software VGStudio Max 3.0 (Volume Graphics GmbH, Germany). We use a common thresholding (ISO-50%) [75] for determining the surface. Fiber analysis tool is utilized for obtaining fiber orientation and fiber volume results for all the simulated scans. A workflow for the analysis is shown in Fig. 2.5.

For fiber orientation, the mode "plane projection" is selected since it is best suitable when injection molding parts with a preferred plane or surface of orientation are analysed. It calculates the 3D orientations of the fibers, projects them into a user-defined plane and then calculate the projected angle from the reference axis within that plane (see Fig. 2.5 c). The fiber orientation tensor is extracted for better interpretation and plotted against the thickness of the part; the results are shown in Fig. 2.6. The tensor has three principal components (A11, A22, A33); only the first principal component (A11) is considered here for the comparison. The results show almost identical behaviour as seen in Fig. 2.6 a; however, there are but very negligible deviations from model values (Fig. 2.6 b and c). Overall, the CT settings and selected resolutions are not significantly affecting the fiber orientation results which is very interesting as the resolution is expected to affect the CT based analyses' results considerably. Furthermore, it is very useful, when a global analysis is required and/or it is not practically possible to achieve very high-resolution due to part size. Nevertheless, it is suggested to examine resolutions lower than eight μm.

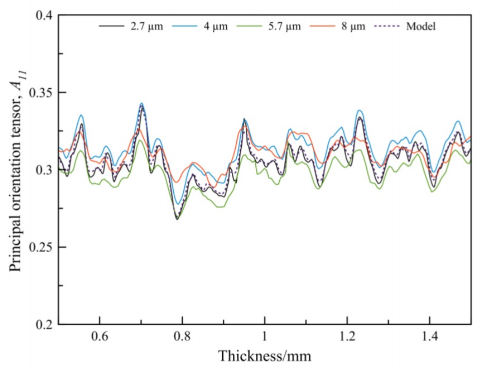
On the other hand, the fiber volume content is greatly affected by the resolution, as visible in Fig. 2.7. However, the effect of CT parameter settings is not so significant similar to the fiber orientation. It becomes more evident in the segmented 2D slices taken at the center of the part (Fig. 2.8) that the fiber content is overestimated at lower magnifications (for the used algorithm of thresholding), which is due to the increased voxel size. Increasing the voxel size (i.e. decreasing the spatial resolution) from 2.7 μm to 8 μm, results in a three times



(a)

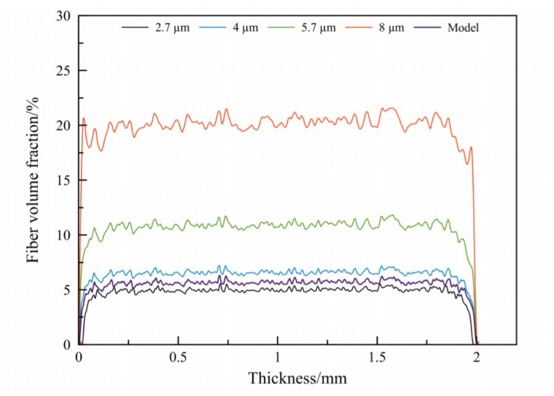


(b)

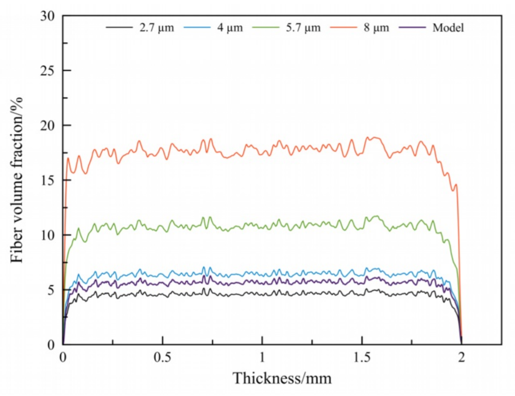


(c)

FIGURE 2.6: Analysis of the orientation tensors. (a) Fiber orientation tensor component for Set-1, (b) the section view for Set-1, and (c) the section view for Set-2.



(a)



(b)

FIGURE 2.7: Analysis of the fiber volume. (a) Fiber volume content as a function of thickness for Set-1, and (b) fiber volume content as a function of thickness for Set-2.

higher estimate in fiber content. In order to get comparable results it is recommended to use a voxel size $\leq 4\ \mu m$. It is also visible from the cross section that the error in volume is mainly due to the overestimation of diameter.

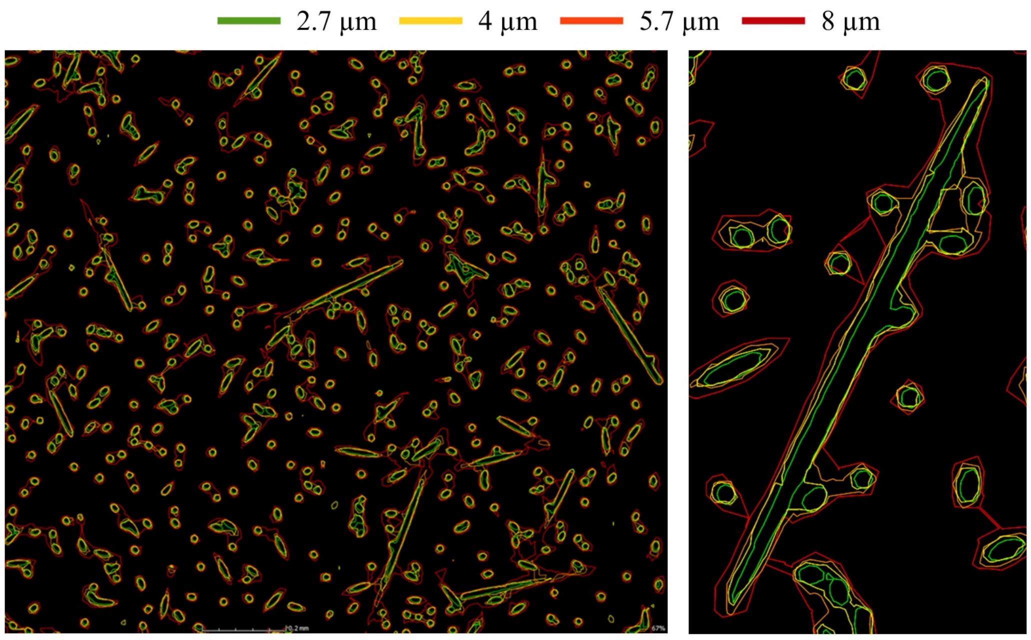


FIGURE 2.8: Visualisation of the segmented fibers at different scanning resolutions. 2D cross-section shows the determined surface (in VGStudioMax) for the fibers at different resolutions in different color.

2.4.2 Experimental results

The manufactured part is scanned in order to validate the simulation results; since both the sets of CT parameters produced almost identical results for the virtual part, only the set-2 parameters are employed for scanning the real part due to its lower scanning time. The part is scanned at all the four magnifications; however, the scanned area for the BR (2.7 μm) and HR (4 μm) scans is partial due to the limited FOV, as explained in Fig. 2.3. Therefore, a common area of (2 $\times$ 2) mm^2 through the thickness is chosen for the analysis of all the magnifications. The area is precisely selected at the center of the part as shown in Fig. 2.9. For injection molded part, it is important to understand the fiber orientation in relation to melt flow direction. The three components of fiber orientation tensor A11, A22 and A33 for manufactured part are shown in Fig. 2.10, which represent the orientation in flow direction, transverse direction and out-of-plane direction, respectively. The major concern here is to see the effect of different CT resolutions on the measured fiber orientation tensor, thus the obtained results for real part are in good agreement with the simulated results, as the resolution is not significantly affecting the fiber orientation. However, the lowest resolution of 8 μm (LR) produces slight deviation in the transverse direction as compared to the identical results of other three resolutions (MR, HR, BR); which could be explained by the fact that further lowering the resolution might produce larger deviations. Therefore, it is concluded that the resolution does not affect the fiber orientation results but up to a certain voxel size.

The fiber volume content results obtained on the real part for different resolutions are shown in Fig. 2.11. The results follow similar trends, which were observed in the simulations. The fiber volume content along the thickness direction is overestimated at a lower resolution. As already stated, the increased voxel size results in significant error in the evaluation of the fiber diameter, which is responsible for the large overestimation of the volume as visible in the cross section in Fig. 2.8. However, a slight underestimation in the volume at the maximal

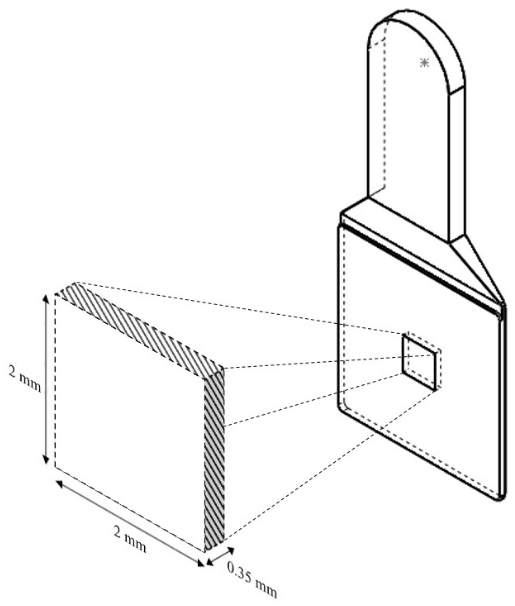


FIGURE 2.9: ROI selection on the real part.

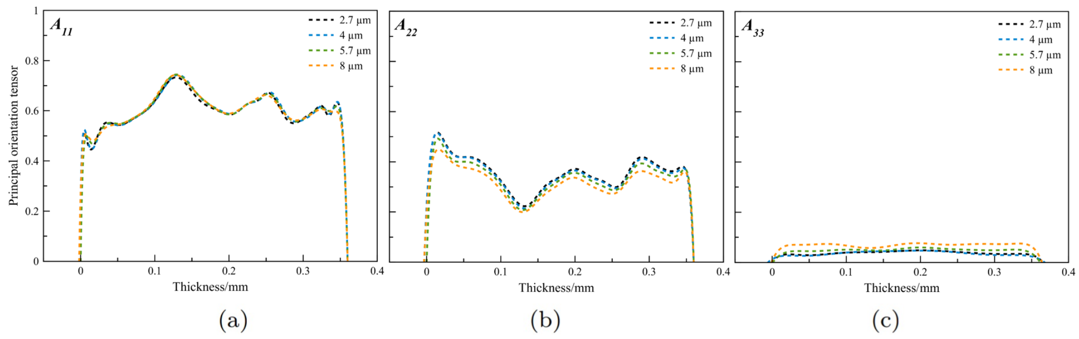


FIGURE 2.10: Fiber orientation results obtained from the real part for all four magnifications: (a) component A11 (along the flow direction) (b) component A22 (along the transverse direction) and (c) component A33 (out-of-plane direction).

resolution is to be noticed in both the simulation and real scans (Fig. 2.12 a). This behaviour can be explained by the increased X-ray image blurring at higher magnification and the resulting shift of the surface determination towards the material side [75].

2.4.3 Open dataset of SRFP CT scans

We provide a public dataset for evaluating (single)-fiber segmentation techniques. The provided scans exhibit typical artifacts and limited resolution. As discussed in the previous chapter we decide to divide the dataset into two main parts with one part containing the real experimental CT scans of SFRP composites and the other containing synthetic CT scans.

At the time of creation of this dataset only MR (3.9 µm) and LR (8.3 µm) scans were available. Because it is impossible for humans to annotate fibers at LR (fibers are too small), we hand-annotated volumes at MR, and use these labels for LR scans after registration. The annotations are further post-processed in order to create the segmentation ground truth.

The synthetic data contains simulated scans of SFRP by using computational model. Therefore, for the synthetic volumes a perfect ground truth is known. Both types of data are

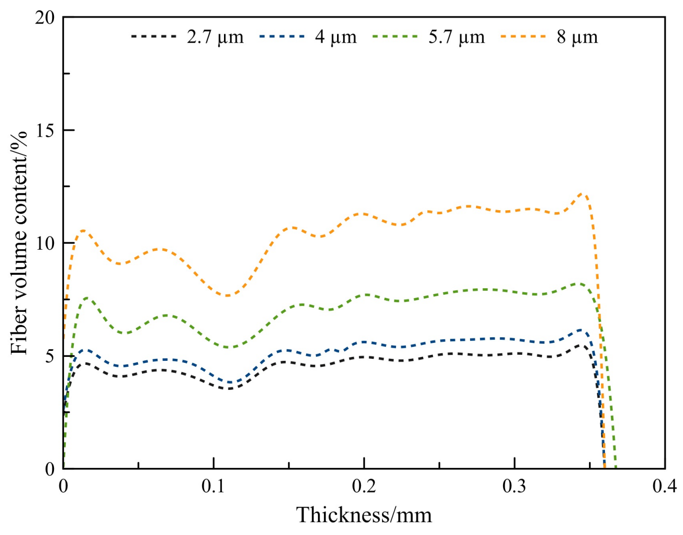


FIGURE 2.11: Fiber volume content results obtained from the real part for all four magnifications.

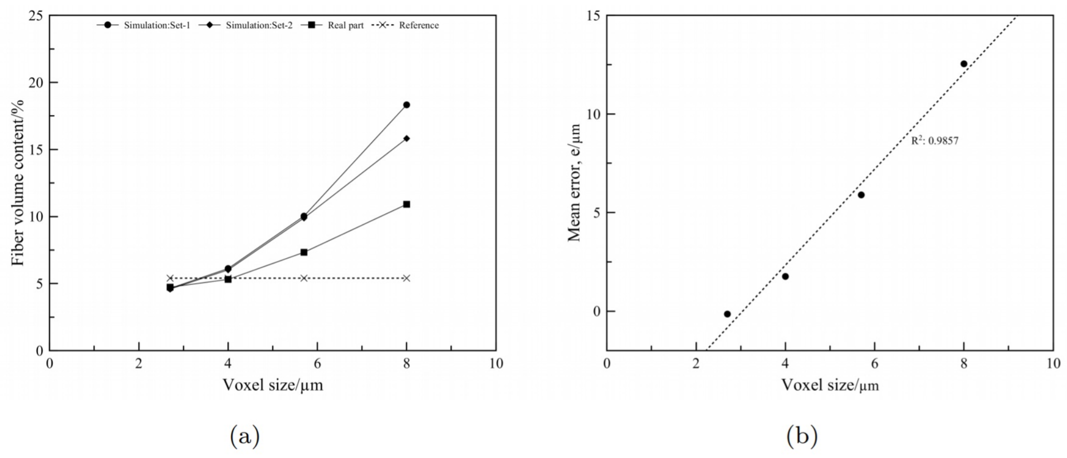


FIGURE 2.12: The obtained fiber volume content as (a) a function of voxel size and (b) the estimated error in diameter as a function of voxel size.

provided with training and test sets with corresponding labels (post-processed hand annotations for the real data and STL models for the simulated data). The annotations are only available for the training datasets, whereas the annotations for the test data are kept hidden from the public. The algorithms may be trained and initially verified on the training dataset, but the final score is computed and evaluated on the organizers' side with the use of the hidden ground truth and labels. Example slices of the data can be seen in Figure 2.13 (a) and (b).

Real experimental data

As described in the previous chapter, the fibers in the material have a diameter of 10 µm to 14 µm and are approx. 1.1 mm in length. The experimental scans are acquired on a Nikon MCT225 X-ray CT system. The scans are performed in both MR and LR with isotropic resolution and the corresponding voxel sizes are 3.9 µm and 8.3 µm respectively. The MR data is further hand annotated by the Knossos labeling tool [49]. During the annotation process,

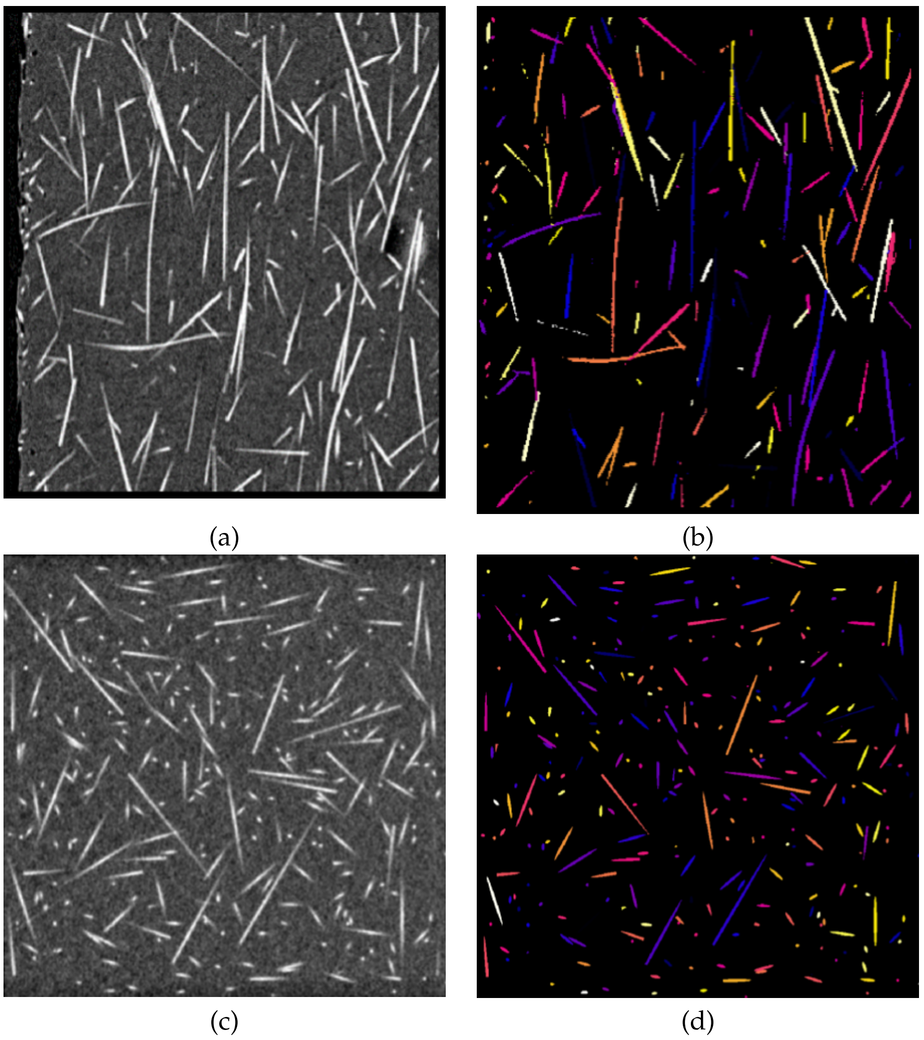


FIGURE 2.13: Example slices of the data together with its corresponding ground truth (the random coloring has been added for visualisation purposes. (a) Slice from a real HR experimental data, (b) corresponding hand made annotated segments after post-processing, (c) slice from a synthetic HR data, (d) corresponding slice of the STL model.

each fiber gets a different index, which is a unique ID number. The annotation process took around one hour per 200 fibers per annotating person.

This includes annotation of particles, which may be used as false positives examples in the future. Each fiber or a particle is annotated as a set of connected points (i.e. one polygonal chain per fiber). Currently we provide 8,000 fiber annotations for two regions of two scans in MR (around 4,000 annotations per volume) and corresponding, registered LR scans.

As the annotations from Knossos are single connected points, we develop a post-processing script to convert it into a volume of annotated segments. The steps of the pipeline are as follows. First, the 3D version of Bresenham's line algorithm [12] is used on the hand-labeled polygonal chains. The algorithm draws straight lines on a 3D grid and renders them into a volume. In the next step the lines are used as seeds for the 3D region growing segmentation algorithm [125] which is applied on the input volume (real volumetric data). The region growing stopping criteria has to be set manually and the value depends on the data. The resulting output is the desired volume of annotated segments. In this way, each experimental scan has a corresponding volume with ground truth voxel labels. That is, the

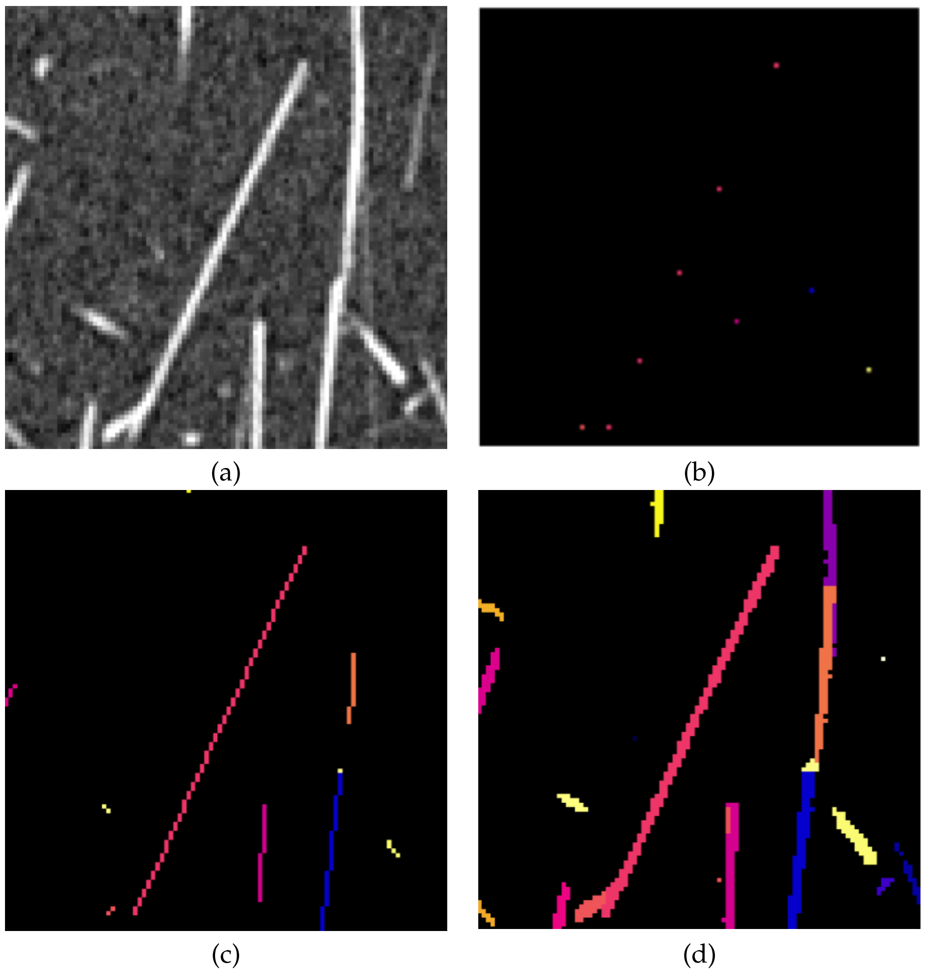


FIGURE 2.14: The post-processing pipeline steps. For the visualisation purpose each fiber has a random color. (a) Zoomed region from a slice of real volumetric data, (b) the Knossos polygonal chains rendered as voxels, (c) rendered lines after 3D Bresenham's line algorithm, (d) rendered annotated segments after 3D region growing.

annotated voxels have corresponding index value for voxels assumed to contain fibers or value 0 for the background (not fibers). Figure 2.14 presents example slices from the pipeline correspondingly. We implement Bresenham's line algorithm in Python and used the VIGRA implementation of the 3D region growing algorithm [67].

Synthetic data

Since the hand-annotated data is limited, we can greatly extend the number of volumes in the data; and the quality of the hand- annotations may raise a discussion. Therefore, adding synthetic scans to this study provides a number of advantages. First, there are more annotated volumes without the need of manual annotations. Second, having the computational model allows us to have the perfect ground truth for the synthetic scans in comparison to hand-annotations on the real scans. The synthetic scans are based on a computational SFRP composite model generator, which has been written by use of an in-house software. We wanted our synthetic scans to have similar parameters to the real scans. The settings of the model is set to mimic the characteristic of the PBT-10%GF material [38]. For the X-ray imaging simulation of a scan, we use the software package aRTist 2 [6]. We design the source and the detector to match the parameters of the Nikon MCT225 X-ray CT system [92]. The proposed computational model of SFRP composite is set to match the fiber content of 5.4±0.5% of the entire volume (that is around 10% weight fraction). We create the SFRP composite model by defining each fiber in the model as a thin cylinder with a fixed radius at a random orientation, position and with a random length. For this work, we set the volume of the composite to be a 2 mm^3 cube. The radius is set to 6.5 µm and the length is sampled from a normal distribution with mean of (500 ± 100) µm (see 2.1). The algorithm for the model creation works in an iterative way. Once the desired dimension of the synthetic volume is defined (the dimensions of a cube), the algorithm tries to fit randomly created fibers inside it. At the end of each successful iteration (i.e. an iteration after which a fiber is added to the model), the fiber ratio is calculated based on the provided densities of the fiber and epoxy material [98]. For instance, in the example in Table 2.5, out of 150,000 randomly generated fibers only 6,628 fit into the cube. The algorithm stops when the desired fiber ratio or the maximum number of attempts is reached. If the fiber has been generated inside the cube, and fits in such a way that it does not overlap with the previous fibers, it is saved as a set of points defining its surface. Because of this fitting process, the fibers are forced to be almost parallel to the surface of the cube the closer they are to its surface. Computation time depends on the properties of the model. The higher the desired content ratio of fibers the longer it takes to create the model. For the properties described above, preparation of one computational model takes around six hours of computational time on single i7-8750 CPU. Example parameters for a single model and resulting statistics are presented in Table 2.5.

For the X-ray imaging simulations, we use the generated STL model of fibers and embeds it inside a cube geometry, which is used as epoxy matrix. The densities of glass fibers (2.54 g/cc) and the surrounding epoxy matrix (1.31 g/cc) are set to the characteristics of the PBT-10%GF material [65] (see 2.1). The synthetic scans are also simulated with isotropic resolution in LR (8.3 µm) and MR (3.9 µm). The source-object distance (SOD) and source-detector distance (SDD) were set to 23.36 mm and 1177.08 mm for the MR and 48.96 mm and 1176.96 mm for the LR respectively. In both MR and LR settings, we use four projections to average and performed in total 2,000 projections per model. The detector and source parameters are the same for LR and MR settings and are similar to the characteristics of the Nikon MCT225 X-ray CT system. The detector size is set to 2,048×2,048 pixels, with pixel resolution of 0.2 mm. The voltage is set to 120 kVp, and the current to 71 µA. For the noise factor, we check the signal to noise ratio (SNR) of the air of a real experimental projection and match it to our simulation. The reconstruction is performed by use of the standard CT filtered back projection (FBP) reconstruction algorithm implemented in the CT Reconstruction module of the commercial software VGStudio Max 3.0 [124]. Fig. 2.13 (c) and (d) show one slice of the synthetic scan and the corresponding STL model. The STL model serves as a ground truth for the resulting volume. In order to get a binary mask the STL models are rendered into binary volumes. The entire simplified process of synthetic volume preparation from the computational STL model is presented in Figure 2.15.

2.5 Conclusion

The major advantage of CT technique is its non-destructive nature, though which is partially true when high resolution is required and the part is larger than the field of view. Thanks to the simulation study, it is evident that the fiber orientation results are not significantly

TABLE 2.5: Synthetic model parameters and resulting STL model statistics.

Resulting model statistics	
Number of fibers	6,628
Average length	0.491 mm
Total fiber volume	0.432 mm^3
Max length of a fiber	0.898mm
Min length of a fiber	0.126mm
Entire volume	8mm^3
Fiber content volume	0.432mm^3
Fiber volume fraction	5.40%

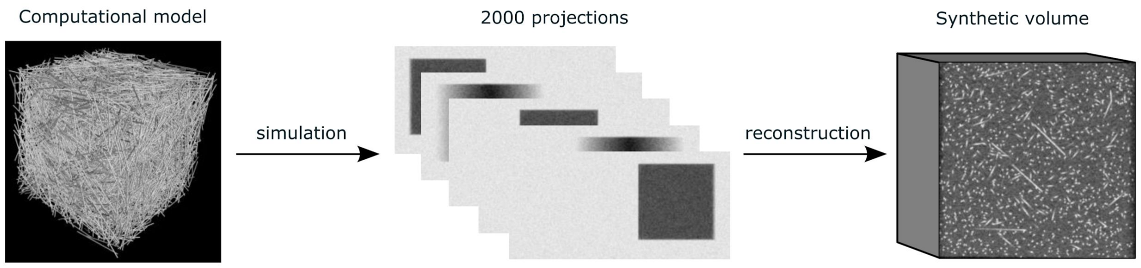


FIGURE 2.15: Sketch of the synthetic volume preparation. The computational model of SFRP in STL format is used in combination with the commercial simulated x-ray imaging software to produce a number of projections. The projections are reconstructed with the standard FBP algorithm implemented in VGStudio Max.

affected by the resolution (for the tested range up to 8 μm) and the employed CT settings. This observation was confirmed by the experimental results from the real part. On the other hand, the fiber volume is greatly affected by the voxel size of the CT data, e.g. 300 % over-estimation of fiber volume for voxel size 8 μm in simulation. Similar results have also been obtained from the experiments on the real part: which show a good agreement with the reference (up to a voxel size of 4 μm) and then large deviations in both the simulated and real part results are witnessed. It was also demonstrated that the error in volume measurement can be compensated with the known fiber length, diameter and number of fibers. Furthermore, it can be concluded that the effect of the spatial resolution on the fiber orientation results is negligible for the voxel sizes up to 8 μm). Therefore, a large area of the part or the full part (as per the requirements) can be considered for the fiber orientation analysis. However, the CT based fiber volume content analysis is very sensitive to the resolution, therefore, the resolution should be chosen carefully.

We provide a dataset of SFRP composite CT scans for quantitative comparison of segmentation techniques together with evaluation metrics. The dataset is designed and prepared in order to evaluate (single)-fiber segmentation algorithms. However, both its real and synthetic part may be used outside the framework of the challenge. For instance, one can use the synthetic data and its corresponding model to evaluate algorithms, which measure the local orientation distribution. For the real part, we design and implement a post-processing pipeline for the Knossos – polygonal chain annotations. For the synthetic part, we design a computational model of SFRP and use it in combination with third-party computer simulated X-ray imaging software. Additionally, we propose a base-line algorithm for the task of fiber segmentation and its implementation. The post- processing pipeline might be extended and used on its own to measure statistics like length or orientation distribution for a small subset area of a CT scan.

Chapter 3

Fully Convolutional Deep Network Architectures for Automatic Short Glass Fiber Semantic Segmentation from CT scans

3.1 Outline

We present the first attempt to perform short glass fiber semantic segmentation from X-ray computed tomography volumetric datasets at medium (3.9 µm isotropic) and low (8.3 µm isotropic) resolution using deep learning architectures. We perform experiments on both synthetic and real CT scans and evaluated deep fully convolutional architectures with both 2D and 3D kernels. Our artificial neural networks outperform existing methods at both medium and low resolution scans. Work from this chapter has been published in the International Conference on Industrial Computed Tomography (iCT) 2018 proceedings [70].

3.2 Introduction

Reliable information about fiber characteristics in short-fiber reinforced polymers (SFRP) is much needed for the process of optimization during the product development phase. The main characteristics of interest are fiber orientation, fiber length distribution and the percent composition in the product. The influence of these characteristics on the mechanical properties of SFRP composites is of particular interest and significance for manufacturers. The recent developments of X-ray computed tomography (CT) for nondestructive quality control enabled the possibility to scan the materials and retrieve the 3D spatial information of SFRPs. In recent years, deep learning methods have revolutionized various fields to which they have been applied. In computer vision, fully convolutional networks (FCN) have become the architecture of choice for the task of semantic segmentation [118]. In particular, FCNs have also been successfully applied to a number of medical 3D X-ray datasets [110]. This motivates us to apply these ideas to the task of fiber semantic segmentation of SFRP. Semantic segmentation is the task of assigning each voxel a label out of a predefined set of categories. The problem of fiber semantic segmentation is therefore a binary classification with 'fiber' and 'non-fiber' categories. This information can be used for further analysis like single fiber segmentation or directly to compute the fiber volume ratio and consequently the fiber weight in a specimen. However, the spatial resolution of a scan is a limiting factor and makes it difficult for the standard segmentation methods to work properly. The currently used methods have difficulties with fibers that are approx. two voxels in diameter or less. The glass fibers we use are approx. 13 µm in diameter. Therefore, we consider scans at a medium resolution (MR) of 3.9 µm, which is a limit for which the standard methods work reasonably well, and a low resolution (LR) of 8.3 µm for which standard methods usually fail. We implement and evaluate a residual [47] version of a FCN with 2D and 3D kernels. Both networks are capable of segmenting fibers better than our baseline algorithms. We also find that training on synthetic volumetric data, but predicting on real scans gives good results. This supports the argument that one does not need a large amount of annotated data to train the networks. We compare our results with Otsu thresholding [96], Hessian based Frangi vesselness measure [36], and our internal implementation of a classical machine learning setup with a random forest [11] using a set of predefined features.

3.2.1 Related Work

One of the easiest and most widely used algorithms for semantic segmentation is Otsu thresholding [96], which is a gray value based histogram method. The reliability of global histogram-based methods is limited due to noise and brightness variations over the image. To overcome this problem, slice-wise circular Hough Transform or Circular Voting [31, 101] can be used. These methods take advantage of the geometrical information about the scanned part, e.g. the radius of a tubular structure, in order to search for circular or elliptical structures in 2D slices. However, these algorithms do not scale well to 3D data. The most common methods to segment tubular-looking structures in 3D data use Hessian based filters. One of the first Hessian based methods, Frangi vesselness filter [36], was initially developed for segmenting vessels in biomedical images and is now commonly used as a preprocessing step on 3D CT data. For fiber-reinforced polymers, a priori information such as the radius of fibers or expected orientation distribution can be incorporated into the algorithm. The method developed by Zauner et al. [135] is a good example of using a priori knowledge dedicated to the analysis of fiber-reinforced polymers.

3.3 Material and Methods

3.3.1 Network Architecture

For this work, as mentioned in the introduction, we decide to use the framework of fully convolutional architecture with residual units. A residual unit consists of two convolutional layers followed by a batch normalization and a non-linearity layer. We do not use pooling layers in our architectures, not to decrease the resolution of already very small fibers. For non-linear functions we decide to use the standard rectifier linear unit. We evaluate a slice-wise 2D and a 3D model. We also compare a relatively shallow deep model and a deeper version of it (with more convolutional layers). We term these models as a "Shallow model" and a "Deep model". The example diagram representation of these models is shown in Figure 3.1.

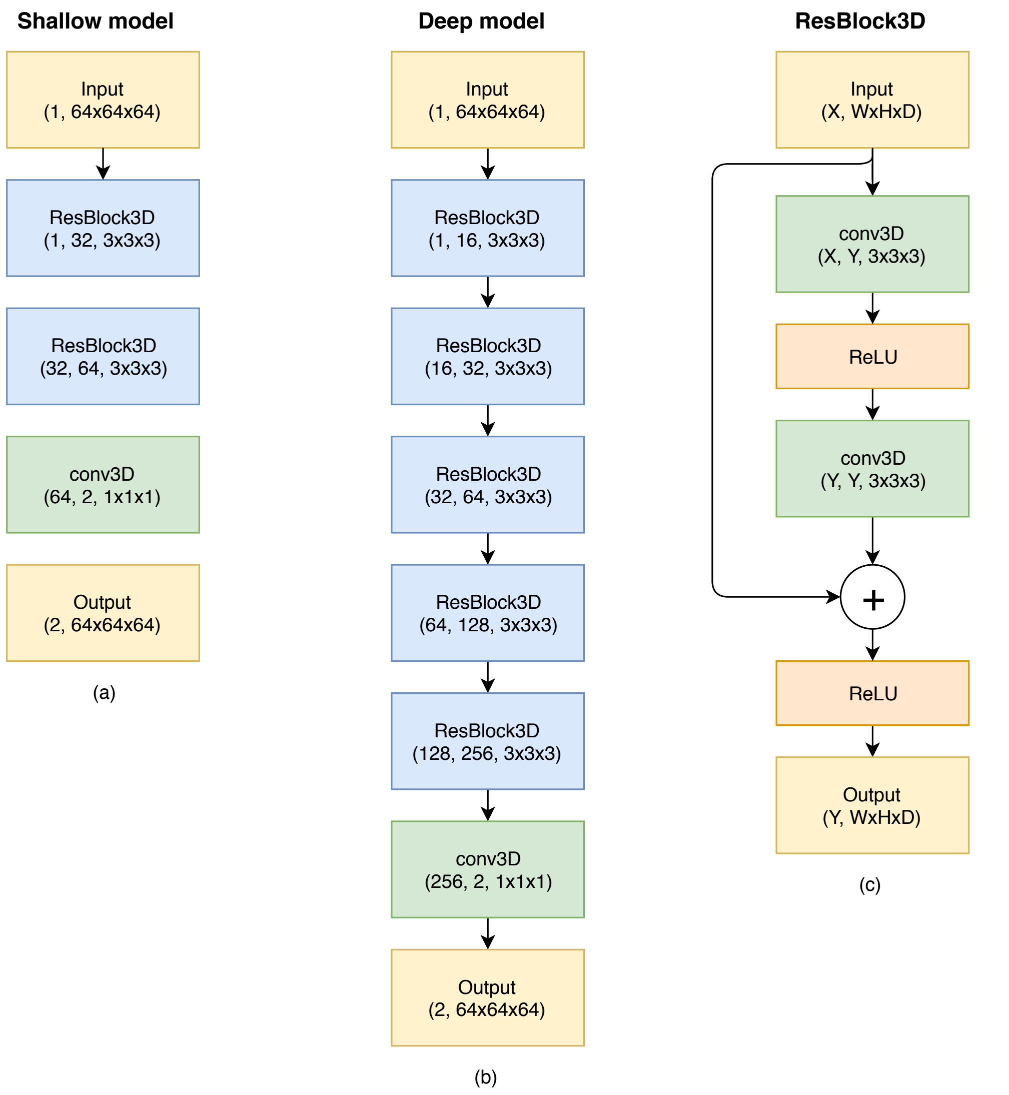


FIGURE 3.1: Diagram representation of the models used for learning from 3D MR patches. (a) Shallow model and (b) Deep model version of the architecture. In both cases the input is a one-channel block of dimension $64 \times 64 \times 64$. Going through the network, the number of channels increases creating a feature maps of the patch. It is done by convolving the input with a number of $3 \times 3 \times \times 3$ kernels (weights) inside the residual blocks (ResBlock3D). The final convolutional layer maps the last feature maps into a two-channel output. One for the fibers, and another for the background (epoxy). c) The architecture of a residual block. It consist of two convolutional layers with a non-linearity in between and an identity connection. X is the number of channels of the input, and Y is the desired number of channels on the output. W, H and D are correspondingly width, height and depth of a patch.

3.3.2 Dataset

We use a publicly available dataset of CT scans of SFRP [69]. The parts from which the dataset was created are manufactured by micro injection molding using PBT-10% GF, a commercial polybutylene terephthalate PBT (BASF, Ultradur B4300 G2) reinforced with short glass fibers (10% in weight). The real scans are acquired by a Nikon MCT225 X-ray CT system. The synthetically modeled scans are generated by our own software. The synthetic scans have the same resolution, same fiber diameter and fiber density as the real scans, but different orientation and length distribution. Finding the length and orientation of fibers is out of the scope of this work. Since the orientation of the fibers in real scans was unknown,

TABLE 3.1: The summary of the data used in this work.

	Resolution	Dimensions [voxel]
Real MR 1	$3.9\mu m$	$200 \times 260 \times 260$
Real MR 2	$3.9\mu m$	$200 \times 260 \times 260$
Synthetic MR 1	$3.9\mu m$	$627 \times 586 \times 594$
Synthetic MR 2	$3.9\mu m$	$635 \times 603 \times 619$
Synthetic LR 1	$8.3\mu m$	$323 \times 340 \times 349$
Synthetic LR 2	$8.3\mu m$	$323 \times 307 \times 424$

we decide to model the synthetic scans with uniformly oriented fibers. We use two synthetic and two real CT scans, where each volume has corresponding binary ground-truth annotations with voxels annotated as fibers. The real scans are provided only in MR, while the synthetic are in both MR and LR. All scans have isotropic resolution. The summary of the data is shown in Table 3.1.

3.4 Results

First we compare the performance of the deep neural networks with the standard methods, then we examine different setups of the model. We evaluate a 2D versus 3D version of the model and the influence of the number of the convolutional layers (residual blocks). For all experiments we use the Dice score as the evaluation metric for quantitative comparisons as proposed in [69].

3.4.1 Training details

All models are limited to 8,000 training iterations with a batch size of three. We use the standard Adam optimizer [64] with a standard learning rate of 0.001. All volumes in the dataset are normalized to have unit variance and zero mean. The patch size of the training data is set arbitrarily, and kept to cover the same region for MR and LR volumes. The patch size is limited by the memory of a GPU board used for training the network, that is why we use smaller patches for the 3D models. The LR models are trained on 32×32 patches in the 2D version and $16\times16\times16$ volumetric patches in the corresponding 3D version, while the MR models are trained on 64×64 patches in the 2D version and $32\times32\times32$ volumetric patches in the corresponding 3D version. For data augmentation, patches are randomly flipped and rotated (by 90°, 180° or 270°) during the training phase. While the training is performed on subpatches of a volume, the evaluation is performed and reported on the entire testing volume. The models are implemented in PyTorch [99] and trained on a single GPU Titan X in Pascal architecture with 3,584 CUDA cores and 12 GB of memory.

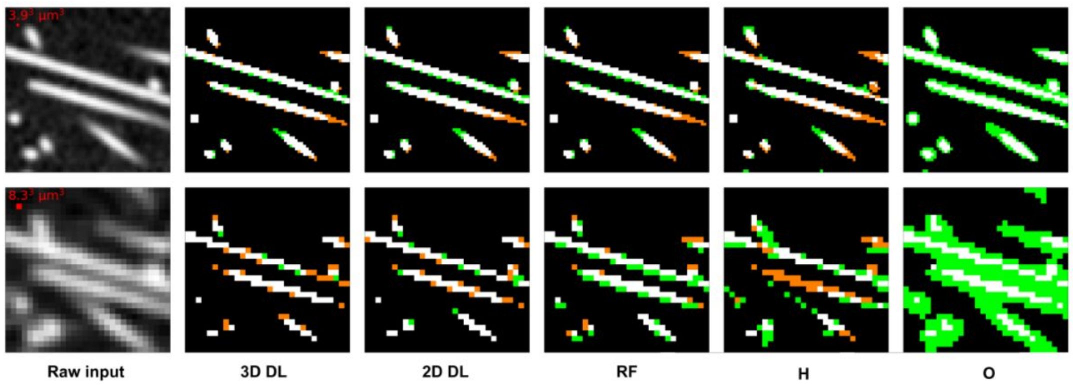


FIGURE 3.2: Visualization of performance of different methods on a synthetic test scan at MR (first row) and LR (second row). The first column shows a cropped region of a slice of the volume. The following columns correspond to the output of the following algorithms: (3D DL) 3D deep learning network, (2D DL) 2D deep learning network, (RF) random forest, (H) vesselnes measure using Hessian eigenvalues, (O) Otsu thresholding. White color means the output matches the fiber (True Positive), black means the voxel is correctly assigned as a polymer (True Negative), green means the algorithm assigned a voxel as a fiber while it was not (False Positive), orange means that a voxel belongs to a fiber, but the algorithm wrongly assigned it as a polymer (False Negative).

3.4.2 Comparison with standard methods

We perform the comparison on one volume (either MR or LR) and evaluate on another one. If the method does not require learning we simply report its performance. We provide the result of finding the best threshold on the particular volume based on the groundtruth, which is the upper boundary for threshold based algorithms. In this experiment we use only the deep models for comparison. An example visual comparison of the methods is shown in Figure 3.2. For the MR data Otsu thresholding, depending if the data is synthetic or real, is close to the optimal threshold (which is not known a priori). Otsu thresholding is sensitive to the histogram distribution of voxel intensities and in a result, because of slightly different statistics of the synthetic volume, it is not able to find a good threshold for it. The Hessian based method works better on the synthetic data than Otsu and worse on real data. We reason that it is so, because fibers in the synthetic data are randomly oriented straight tubes, and therefore have less touching points compared with real data. Random forest, which uses a set of standard features (Gaussian smoothing, Gaussian gradient magnitude, Laplacian of Gaussian, eigenvalues of Hessian of Gaussian and eigenvalues of structure tensor) produces better results than the Hessian based technique and works better for the synthetic data than Otsu. Deep neural networks outperform the other methods in terms of Dice score at both MR and LR settings when trained on a volume with similar statistics. Especially at LR, deep models achieve a much better result. Deep neural networks perform similarly to Otsu best performance in terms of Dice score for the real MR 1 scan when trained on a synthetic volume. The 3D version of the deep model is working as well as the 2D slice-wise version for MR data, but is significantly superior at LR data.

Results gets more interesting for the LR data. Even the best possible threshold is far from being acceptable. As a result, the Otsu thresholding performs the worst. The Hessian based method has difficulties at low resolution, because fibers are getting closer to each other as the resolution decreases and therefore have more touching points. The trainable random forest works better than both Otsu and Hessian, but it is not able to find as good a representation

TABLE 3.2: Performance of various methods. We compare the Otsu thresholding (O), Hessian based filtering (H), Random Forest (RF), proposed 2D Deep Learning (2D DL) and 3D Deep Learning (3D DL) approach. "R" and "S" next to either MR or LR stands for "real" and "synthetic" data, respectively. The trainable methods were learned on Real MR 1, Synthetic MR 1 or Synthetic LR 1. Results in Dice score. The best threshold is known in advance from the groundtruth and is provided as the upper boundary for the Otsu thresholding. The highest achieved Dice scores for particular evaluation volumes are bolded.

			Dice score of prediction				
Evaluation volume	Training volume	Best threshold	O	H	RF	2D DL	3D DL
R MR 2	R MR 1	0.993	0.892	0.671	0.957	**0.986**	0.979
	S MR 1				0.796	**0.903**	0.881
S MR 2	S MR 1	0.875	0.657	0.767	0.857	0.885	**0.898**
S LR 2	S LR 1	0.505	0.289	0.374	0.536	0.750	**0.837**

as neural networks. Random forests are limited by features chosen and hand-designed beforehand. Neural networks are clearly outperforming the rest. See the results for both MR and LR in Table 3.2.

3.4.3 Variations of the deep learning architecture

We compare four slightly different setups of deep networks. We train the shallow and the deep version of the deep residual network architecture in both 2D and 3D variants (see Figure 3.1) on real MR 1, synthetic MR 1 and synthetic LR 1 and evaluate on the other corresponding volumes. We discuss the performance and the capability of a network to generalize. The results are in Table 3.3.

3.4.4 Highest score

To achieve the highest Dice score one has to train the network on an identical volume with similar statistics to the evaluation volume. For example, regardless the architecture, all networks achieve a Dice score of approx. 0.980 when trained on Real MR 1 and evaluated on Real MR 2. This is considerably better compared with standard methods. The same holds for training on Synthetic MR 1 and evaluating on Synthetic MR 2. When evaluating on the MR data, it seems that it is enough to use a 2D version of a network, with the 3D version not always working better. The deep version is usually only very slightly increasing the performance. Might be that the resolution is high enough that simple low level features already provide a high accuracy prediction. It is also easier to train it in a 2D setup, because of a lower number of trainable parameters. For LR data it is no longer the case. We find that already the shallow 2D version of the deep network is performing better than the other standard methods at LR. The deep 3D version is achieving the highest Dice score among the other architectures. This means, that accurately processing short-glass fiber data at LR requires a richer feature representation.

TABLE 3.3: Comparison of different deep learning setups. Trained on real MR 1, synthetic MR 1 and LR 1. Results in Dice score. The highest achieved Dice scores for particular pair of evaluation and training volumes are bolded. "R" and "S" next to either MR or LR stands for "real" and "synthetic" data, respectively.

		Dice score of prediction			
Evaluation volume	Training volume	Shallow 2D	Deep 2D	Shallow 3D	Deep 3D
R MR 2	R MR 1	0.980	**0.986**	0.983	0.979
	S MR 1	**0.928**	0.903	0.876	0.881
S MR 2	R MR 1	0.775	0.774	**0.801**	0.788
	S MR 1	0.875	0.885	0.895	**0.898**
R MR 1	S MR 1	**0.929**	0.904	0.873	0.885
S MR 1	R MR 1	0.787	0.789	**0.818**	0.802
S LR 2	S LR 1	0.630	0.750	0.803	**0.837**

3.4.5 Generalization

To evaluate the network capabilities to generalize we train the network on volumes with different statistics than the evaluation volume. When trained on Synthetic MR 1 and evaluated on Real MR 1 or 2 the scores are still higher or similar to standard methods. The network trained on Synthetic MR 1 generalizes much better compared to training on Real MR 1 (and evaluate on Synthetic MR 1 or 2). From that we conclude that when one does not know the statistics of the evaluation volume it is better to train the network on synthetic data than on real. Networks seem to overfit to certain directions when trained on Real MR 1, while fibers in our Synthetic MR data are uniformly oriented. When trained on Real MR 1 and evaluated on Synthetic MR 1 we can see that the 3D version is better at generalization. From this we conclude that the 3D features are more general, and harder to overfit. We do not note a gain in performance by using a deeper version of the 3D architecture. Unfortunately, at the time of writing this chapter we do not have manually annotated data for real LR so we could not have done the same for real scans. Because of the same reason we cannot evaluate how well the network generalizes to real volumes (with different statistics).

3.4.6 Time evaluation

Lastly, we take a look at the computational effort. We use single Nvidia GPU Titan X in Pascal architecture. Training and evaluation time for a 2D is considerably shorter than 3D. The same holds for deep vs. shallow architecture. For the 2D architecture, the network is predicting one slice at a time. In the 3D setup the network is predicting one patch at a time. These patches are overlapping and the final output is a mean output of patches. These timings could be optimized but it is out of scope of this thesis. See Table 3.4 for example timings for synthetic MR data.

TABLE 3.4: Time table of computational effort comparison of training and evaluation. Trained on a Synthetic MR 1 with 8,000 iterations and 3 patches per batch. It is evaluated on Synthetic MR 2 ($635 \times 603 \times 619$ [voxel]).

	Shallow 2D	Deep 2D	Shallow 3D	Deep 3D
Training	173 [s]	228 [s]	400 [s]	2,467 [s]
Evaluation	17 [s]	49 [s]	184 [s]	1,183 [s]

3.5 Conclusion

We show that deep neural networks outperform other methods at LR and MR when properly trained and achieve the state- of-the-art. In the case of LR data, deep neural network is the only technique producing accurate results. In contrast to Otsu thresholding they are robust to changes in the histogram of intensity, and should produce results at least as good as Otsu. The benefit of using them in comparison with standard machine learning algorithms is the fact that they are capable of adapting the learnable features to the data. We also demonstrate that the network does not require a lot of real hand-annotated data in order to learn a working representation. It is possible to learn the network entirely on a synthetic volume. That means, if one does not want to spend time on annotating data, it is safe to use synthetic volumes at the cost of accuracy in segmentation and still have a higher accuracy than the standard methods.

Chapter 4

Instance Segmentation of Fibers from Low Resolution CT Scans via 3D Deep Embedding Learning

4.1 Outline

We propose a novel approach for automatic extraction (instance segmentation) of fibers from low resolution 3D X-ray computed tomography scans of short glass fiber reinforced polymers. We design a 3D instance segmentation architecture built upon a deep fully convolutional network for semantic segmentation with an extra output for embedding learning. We show that the embedding learning is capable of learning a mapping of voxels to an embedded space in which a standard clustering algorithm can be used to distinguish between different instances of an object in a volume. In addition, we discuss a merging post-processing method which makes it possible to process volumes of any size. The proposed 3D instance segmentation network together with our merging algorithm is the first known to authors knowledge procedure that produces results good enough, that they can be used for further analysis of medium resolution fiber composites CT scans. Work from this chapter is published in the British Machine Vision Conference (BMVC) 2018 proceedings [71].

4.2 Introduction

As discussed in the previous chapter, the methods currently in use are usually based on hand designed features. Since fibers can be described as long tubular objects, the most widely used family of fully-automatic methods is based on Hessian eigenvalues, like the Frangi vesselness filter [36]. Using a set of Hessian based filters at a number of scales, a confidence map of fiber occurrence can be produced.

4.2.1 Related work

To extract individual fibers, a template matching [101] [31] or a watershed splitting and skeletonisation technique [117] [136] is then applied. However, the performance of these

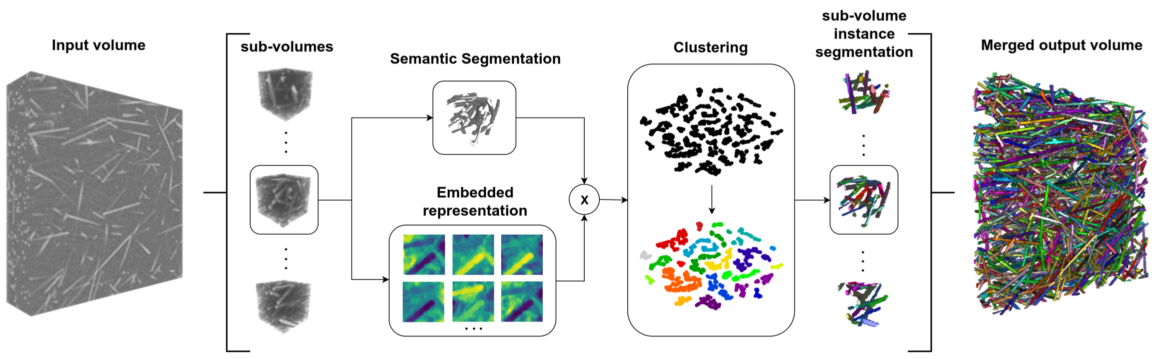


FIGURE 4.1: Sketch of the proposed method. The network is processing overlapping sub-volumes of the input volume. For each sub-volume a semantic segmentation mask and an embedding is produced by a deep network. A clustering method is then applied on the segmented regions of the embedding representation producing clusters corresponding to individual fibers. Fibers are then mapped back to the spatial domain. The overlapping instance sub-volumes are then merged into an output volume.

methods degrades severely if the resolution is too low and fails to produce meaningful results [69]. We show a deep learning based method superiority over Hessian based techniques to produce more accurate results for semantic segmentation of fibers at low CT resolution [70]. In this chapter we tackle the problem of instance semantic segmentation, or single-fiber segmentation.

Deep learning architectures are successfully applied to semantic segmentation problems for both natural 2D images and 3D CT volumes [118] [19]. Similar solutions are found for the problem of 2D instance segmentation. Faster R-CNN [106] and the Mask R-CNN [46] architectures are examples of region-proposal-based techniques which are the state-of-the-art for common scene-understanding datasets like COCO [79] or ImageNet [23]. However, it is not clear how this approach can be extended to 3D volumetric data with densely packed objects like fibers in SFRP. This is why for our 3D problem, we opt for alternative deep learning methods for instance segmentation. There are numerous works in which authors try to come up with different ideas for 2D datasets. An interesting idea that could be extended to 3D volumes is proposed by [4] to reformulate the problem of instance segmentation into learning a mapping to watershed energy. Then, for the final output, a Watershed transform is applied to get the instances. Unfortunately, this method is not applicable to our problem, because fibers are usually too thin to have a border. Another promising idea proposed by [109] is to combine convolutional neural networks (CNN) with recurrent neural networks (RNN). The recurrent structures can be then used to keep track of objects that have already been found, and exclude these regions from further analysis by the algorithm.

In this work we propose a novel deep learning architecture for automatic extraction (instance segmentation) of fibers from low resolution (or even medium in our notation) 3D X-ray computed tomography scans of short glass fiber reinforced polymers. The sketch of the method is presented in Fig. 4.1.

We explore and discuss the performance of the presented method achieved by training on a low resolution SFRP CT scan and compare it to a standard watershed splitting and skeletonisation technique. We test the importance of the semantic segmentation branch by replacing it with a ground truth semantic segmentation. To the best of our knowledge, this is the first attempt of using deep embedding learning for the task of instance segmentation on a

3D volumetric data. The proposed method is also the first to successfully retrieve single-fiber segmentation from a low resolution SFRP CT scan, while the outcome of the standard methods is producing unacceptable results. We base our method on an embedding learning approach [129] [10]. The idea is to use a special embedding layer which is placed at the end of a given deep network. The network is then trained by using a special loss function on the final embedding layer which encourages special structure in the embedding space: pixels belonging to the same class should be close, whereas pixels belonging to different classes should be far apart (in the Euclidean metric of the embedding space). The method as a learnable loss function has been first mentioned by [129], and was then used with some modifications in the deep learning architecture of [10] and [32]. These methods achieved competitive performance on 2D datasets compared with the R-CNN based state-of-the-art.

In the problem of fiber segmentation, the network learns a mapping of each voxel and its surrounding from the input to an embedding space in which voxels belonging to one fiber are separated from voxels belonging to another. Unfortunately, there is one drawback to this method. Such a network is capable of processing only one small sub-volume of a volume at a time because of memory limitation. Each time the network processes a sub-volume it assigns an *arbitrary* index to a fiber region. Because of that, we cannot perform a simple merge as it is usually done for a semantic segmentation problem, where the output is a probability of being an object of a certain class. For a semantic segmentation mask one can take a simple average over overlapping regions in order to merge sub-volumes into a full volume. Therefore, in order to produce an instance segmentation for a full CT scan, we propose a post-processing algorithm, which merges the overlapping predictions of small blocks into a consistent prediction for the entire CT scan during the prediction phase.

4.3 Material and Methods

Similar to the work of [10], we extend the Fully Convolutional Network (FCN) architecture [118] designed for semantic segmentation tasks to produce embeddings by using an extra output. The extra output could be attached at the very end of the backbone of the semantic segmentation network as in [91], but in our setup we decide to use two sub-networks. One is responsible for computing the semantic segmentation mask, and the other for computing the embedding of voxels. The network can be trained separately for embedding and semantic segmentation using corresponding outputs or trained together for both tasks at the same time. We refer to the two sub-networks as *semantic segmentation branch* and *embedding learning branch*. The semantic segmentation branch outputs a confidence map that a given voxel belongs to any fiber or not. The embedding learning branch outputs voxels coded in the embedding space. During the training phase, the architecture is trained only based on outputs from the semantic segmentation branch and the embedding learning branch using specified loss functions.

During the prediction phase a clustering step generates clusters corresponding to individual fibers. The clustering method is applied to the embedded voxels which have a high confidence of being a fiber based on the output from the semantic segmentation branch. The outputs from the two branches and the region on which the clustering is computed are presented in Fig 4.2. The clusters are then mapped back to the spatial domain creating a label volume, where each voxel is assigned an integer label corresponding to the fiber instance it

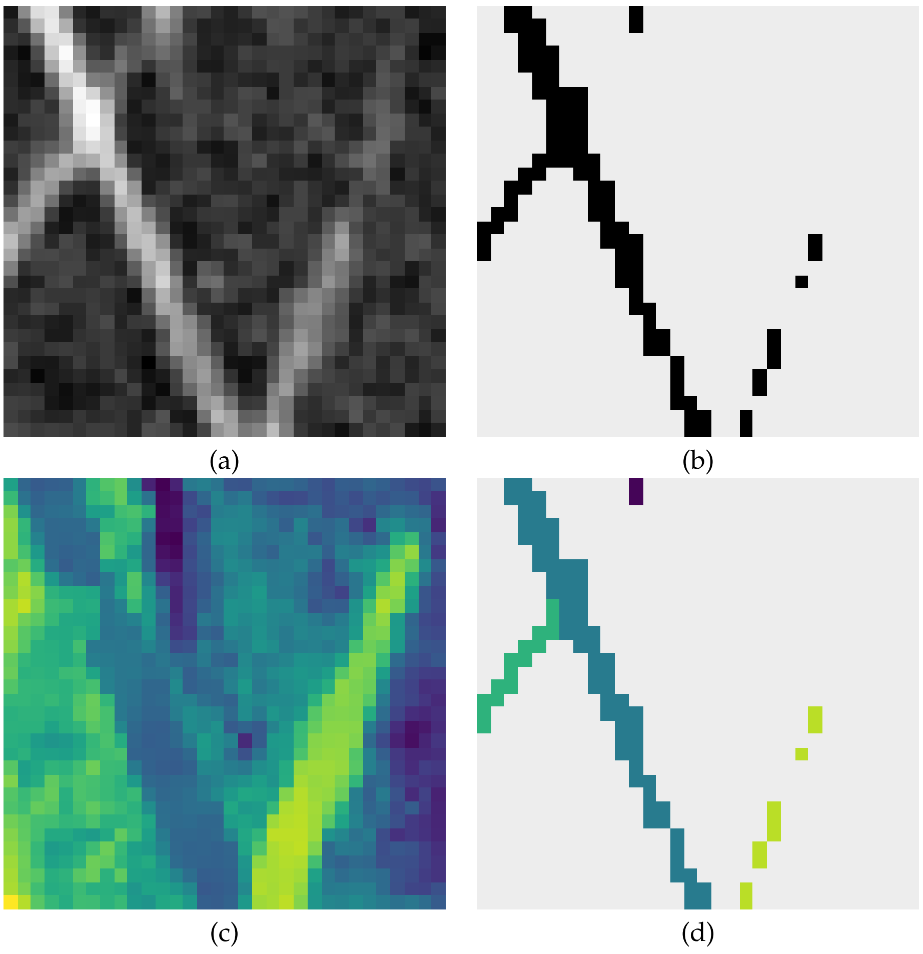


FIGURE 4.2: Visualization of the network outputs. (a) Slice of an input volume patch. (b) Corresponding slice from the semantic segmentation output. (c) Corresponding slice of one (out of many) feature map of the embedding output. The network learns to assign different, unique values (colors) to individual fibers. In this particular feature map all fibers are well separated. (d) The masked embedding by a semantic prediction for which a loss for embedding learning is computed. In the prediction phase it is also the input for the clustering step.

is a part of. To make it possible to use on volumes of any size, we propose a greedy merging algorithm. The network produces outputs for overlapping sub-volumes of the input volume, which are then merged to a full volume. The detailed architecture of the network is presented in Fig. 4.3. In the following sections we describe the above steps in more detail.

4.3.1 Semantic Segmentation

The semantic segmentation branch is a standard FCN for semantic segmentation. We use an architecture that is designed for the task of semantic fiber segmentation [70]. The output of the branch is penalized by the standard voxel-wise binary cross entropy loss L_{CE}, as is common for semantic segmentation tasks. It is defined as:

$$L_{CE} = -[y \cdot log(\hat{y}) + (1 - y) \cdot log(1 - \hat{y})] \tag{4.1}$$

where y are the true binary labels, and $\hat{y}$ are the predicted labels. During the prediction phase, the output is thresholded at value 0.5 in order to produce binary masks. An example slice of an output of the branch is shown in Fig 4.2 (b).

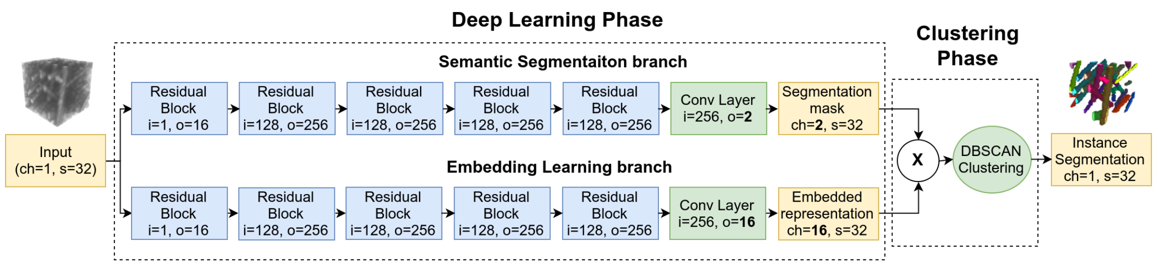

Deep Learning Phase
Semantic Segmentaiton branch
Input (ch=1, s=32)
Residual Block i=1, o=16
Residual Block i=128, o=256
Residual Block i=128, o=256
Residual Block i=128, o=256
Residual Block i=128, o=256
Conv Layer i=256, o=2
Segmentation mask ch=2, s=32
Embedding Learning branch
Residual Block i=1, o=16
Residual Block i=128, o=256
Residual Block i=128, o=256
Residual Block i=128, o=256
Residual Block i=128, o=256
Conv Layer i=256, o=16
Embedded representation ch=16, s=32
Clustering Phase
X
DBSCAN Clustering
Instance Segmentation ch=1, s=32

FIGURE 4.3: Detailed architecture of our network. *ch* is the number of channels of the sub-volume, *s* is the size of the sub-volume (here $s = 32$ means a sub-volume is of size $32 \times 32 \times 32$), *i* is the number of input channels, whereas *o* is the number of output channels from a convolutional layer or residual block. Both residual blocks and convolutional layers use 3D kernels. The kernel size is set to 3 for residual blocks and 1 for the final convolutional layer. Training is performed only on the deep learning phase. During prediction, the output from the embedding branch is masked by the output of the semantic segmentation branch and is processed by the DBSCAN algorithm producing the instance segmentation. The cross sign indicates the masking operation.

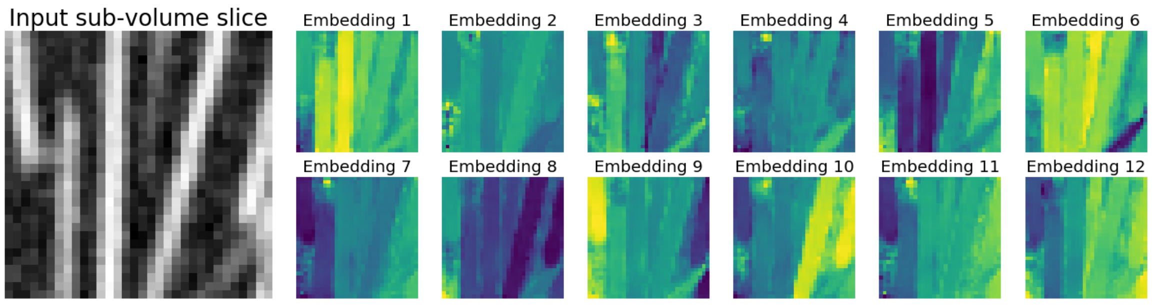


FIGURE 4.4: The input volume is represented by a number of embedded volumes at the embedding branch output. Here, slices of the first 12 embedding volumes corresponding to the input sub-volume slice are visualized. Note, that a good embedding will assign different set of colors in each embedding volume so that the clustering in the embedding space will be easy.

4.3.2 Embedding Learning Loss

The output of the embedding branch is a representation of the sub-volume in an embedding space. The architecture of the branch is identical to the semantic segmentation branch. The only difference is the number of output channels in the final convolutional layer and the loss function. In the semantic segmentation task, the output is producing a volume with two channels, where one is reasoning on the foreground and the other on the background. In the embedding learning, the output has as many channels as the dimensionality of the embedding space (a hyperparameter in the algorithm). An example visualizing feature maps of the embeddings is shown in Fig. 4.4.

The loss function penalizes voxels of different instances that are too close to each other in the embedding space and encourages voxels of the same instance to be close. As a result, the network maps the voxels into the embedding space, such that voxels that belong to the same fiber should be placed next to each other and form easily separable clusters.

We find that the loss function introduced by [10] inspired by work of [129] and extended to 3D by us works best for our problem. Even though we extend the problem to 3D, and use data that contains a large number of objects compared to common scene-understanding problems, the method does not seem to be affected by that. The loss consists of three terms: L_v keeps voxels belonging to the same object close to each other, L_d which forces a minimal distance between clusters of different objects, and L_r which regularizes the cluster centers to be close to the origin. The terms are defined as:

$$L_v = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}[||\mu_c - x_i|| - \delta_v]_+^2 \tag{4.2}$$

$$L_d = \frac{1}{C(C-1)}\sum_{c_A=1}^{C}\sum_{c_B=1, c_A \neq c_B}^{C}[\delta_d - ||\mu_{c_A} - \mu_{c_B}||]_+^2 \tag{4.3}$$

$$L_r = \frac{1}{C}\sum_{c=1}^{C}||\mu_c|| \tag{4.4}$$

where C is the number of objects in the ground truth patch (clusters), N_c is the number of voxels that corresponds to the object c, x_i is the embedding in the final embedding layer, μ_c is the mean of the embedding of object c, $||\cdot||$ is the L_2 norm, and $[x]_+ = max(0, x)$. The parameters δ_v and δ_d are used to control the desired positions of the clusters. The final loss for the embedding learning L_{embd} is a sum of the previous components.

$$L_{embd} = \alpha L_v + \beta L_d + \gamma L_r \tag{4.5}$$

where α, β and γ control the strength of the corresponding term. An example slice of an output of the branch is visualized in Fig 4.2 (c). Note, that the loss is computed only based on the voxels that belong to the foreground fibers. It is the task of the semantic segmentation branch to find the correct position of the fibers.

4.3.3 Clustering

As discussed in the previous section, the semantic segmentation output creates a confidence map that a given voxel belongs to any fiber or not. A clustering is then applied to the embedded voxels with a high confidence of being fibers. An example input slice of one of the feature maps of the embedding is shown in Fig 4.2 (d). In this work, we find DBSCAN [27] to work best on the SFRP dataset. In contrast to Mean Shift used in [10], DBSCAN does not make assumptions about the shape of the clusters. We apply clustering only in the prediction phase because the instance segmentation loss function does not require the instance segmentation map. Note, that DBSCAN does not necessarily assign a label to all voxels. Voxels that are not assigned to any label are assigned as outliers. Clusters are then mapped back to the spatial domain creating an instance segmentation map. Outliers are extrapolated based on their neighborhood in the spatial domain by use of the watershed algorithm, using the clustering labels as seeds. An example visualization of the described steps with help of the t-SNE [81] is shown in Fig 4.5.

4.3.4 Merging

Finally, the inference is produced on small overlapping sub-volumes of the entire volume. Each sub-volume contains different label IDs for fibers, making it not clear which fiber is which. To overcome this problem we design a merging algorithm, which joins label IDs among the sub-volumes based on a spatial distance of fibers in the overlapped regions. The algorithm is applied at each sub-volume and processes recursively one fiber at a time, looking at neighboring sub-volumes with overlapping regions with objects being close to the fiber of interest. The merging procedure is described in more details in Algorithm 1.

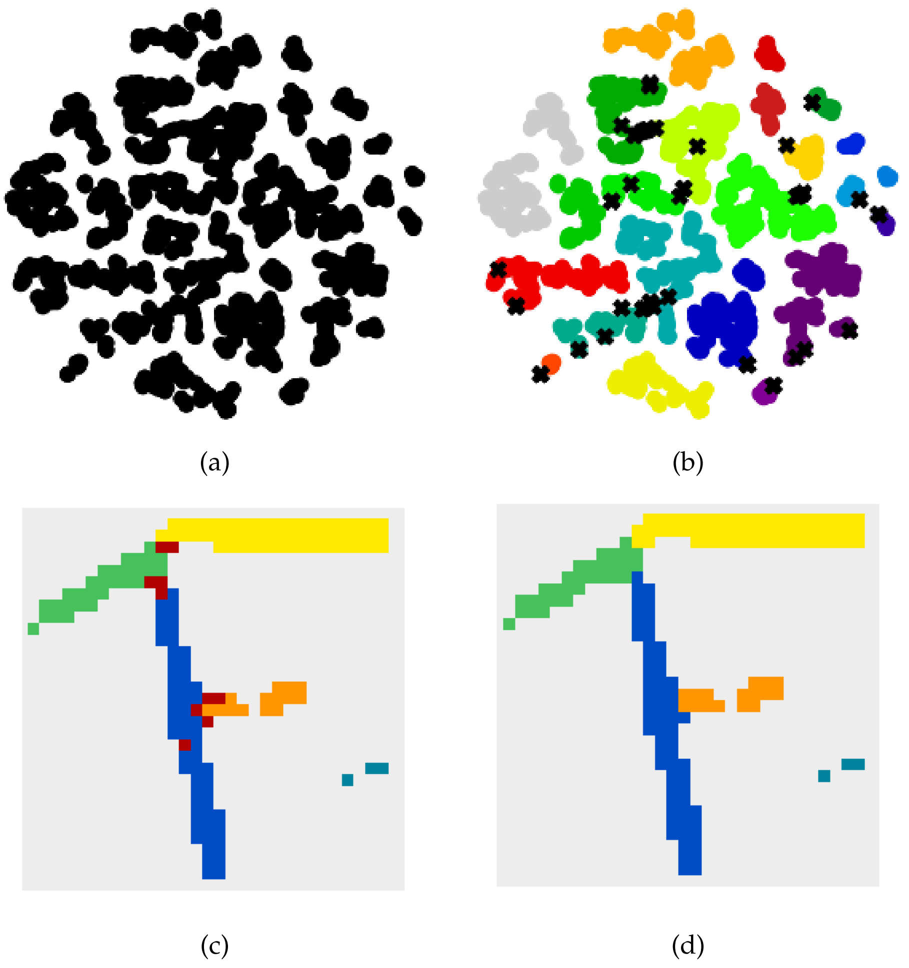

(a) (b) (c) (d)

FIGURE 4.5: 1. Visualization of the clustering steps of the method. (a) Masked embeddings form clusters in a multi-dimensional embedding space (visualized by t-SNE). (b) DBSCAN clusters the embedding representation and assign a different index (color) to each fiber (cluster) with black crosses for outliers. (c) Clusters are then mapped back to the spatial domain. Here a corresponding example slice of the mapping with red pixels for outliers. (d) The watershed algorithm is then applied as a post-processing step to fill the outliers.

4.3.5 Dataset

We evaluate the proposed setup on two hand-annotated regions of MR CT scans of SFRP composites acquired by a Nikon MCT225 X-ray CT system from chapter 2. Scans exhibit typical artifacts and have low, but isotropic resolution. The parts from which the scans were acquired were manufactured by micro injection molding using PBT-10% GF, a commercial polybutylene terephthalate PBT (BASF, Ultradur B4300 G2) reinforced with short glass fibers (10% in weight). The volumes have been hand annotated with center lines and processed by a watershed algorithm to create the instance segmentation ground truth. Both volumes are cubes of dimension $62 \times 260 \times 260$ with approx. 6,500 fibers each. Fibers have a diameter of 10-14 µm (2-3 voxels) and are approx. 1.1 mm long. One scan is used for training, while the other is only used for testing.

Algorithm 1 Merging algorithm

```
1: procedure MERGE($f, p$)                                  ▷ for a fiber $f$ in a sub-volume $p$
2:     $N \leftarrow$ neighbour sub-volume of $p$        ▷ $N$ is a set of sub-volumes neighbouring with $p$
3:     for sub-volume $n$ in $N$ do
4:         $G \leftarrow$ fibers in $n$                            ▷ $G$ is a set of fibers in a patch $n$
5:         for fiber $g$ in $G$ do
6:             $d \leftarrow D(f, g)$                               ▷ Spatial distance between $f$ and $g$
7:             if $d > \alpha$ then
8:                 $g_{id} \leftarrow f_{id}$
9:                 MERGE($g, n$)
```

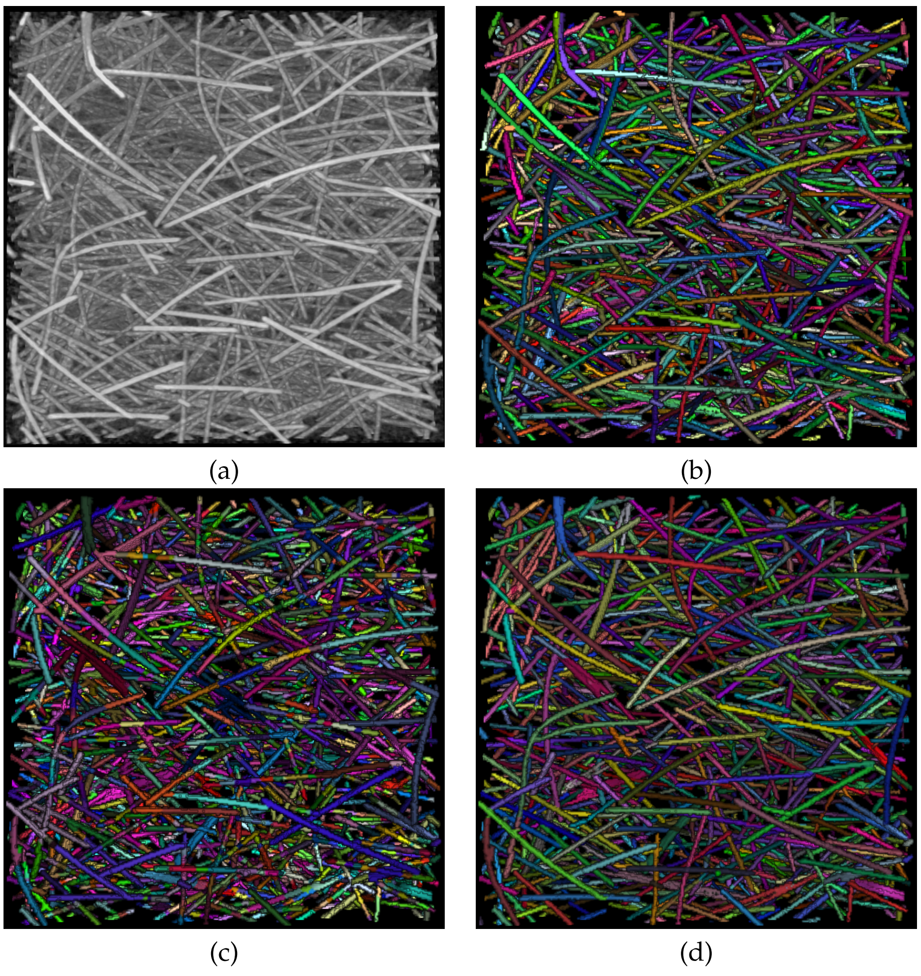


FIGURE 4.6: Visualization of the 3D rendering of the testing volume and corresponding results. The figure presents: (a) the input raw test volume with a certain threshold to remove the epoxy background in the 3D rendering, (b) the corresponding ground truth, (c) the output of a standard connected component (CC) analysis and (d) the output of our method. Fibers are colored semi-randomly based on the fiber ID.

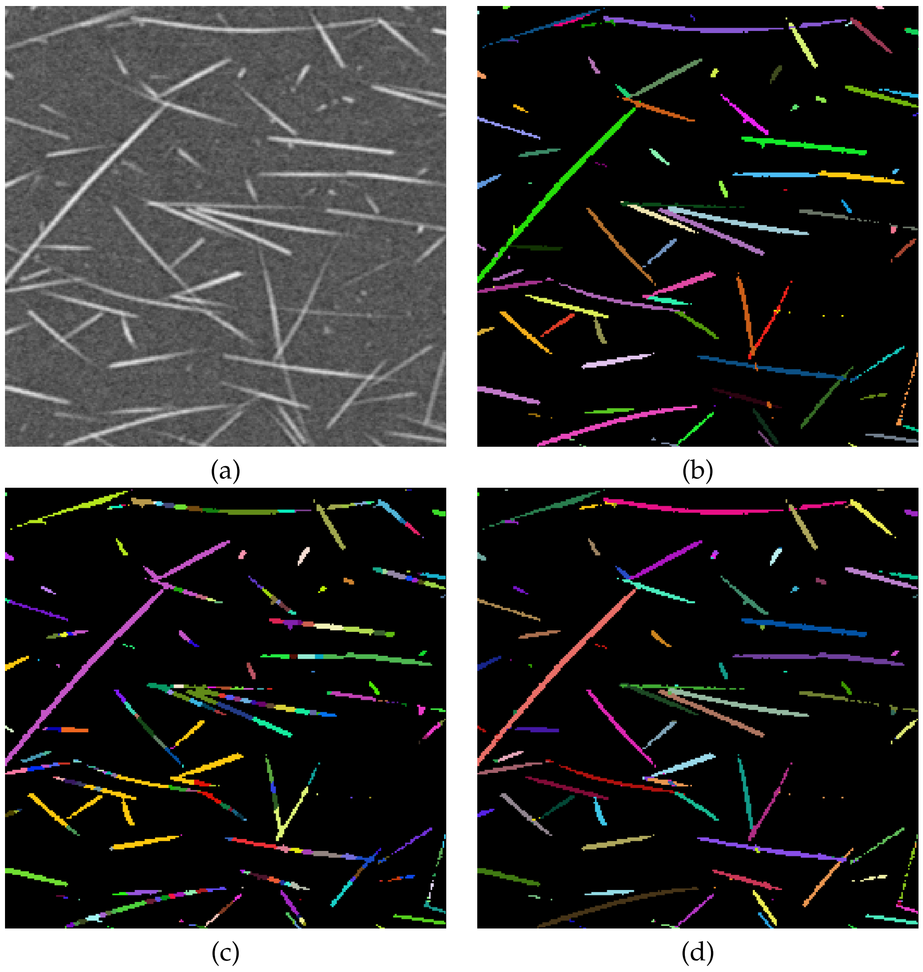

(a) (b) (c) (d)

FIGURE 4.7: Visualization of an example slice of the testing volume and corresponding results. The figure presents: (a) a raw slice of the input test volume, (b) the corresponding ground truth, (c) the output of a standard connected component (CC) analysis and (d) the output of our method. Fibers are colored semi-randomly based on the fiber ID.

TABLE 4.1: Comparison of our method with traditional connected components with and without provided ground truth semantic segmentation mask. Mean ARI are mean values over overlapping sub-volumes of the validation volume, while Merged ARI is the score computed over the entire volume after the post-processing merging step. The Dice score of the semantic segmentation mask from the semantic segmentation branch is 0.9784.

Setup	**Mean ARI**	**Merged ARI**
Embedding Learning	0.9048	0.6529
Embedding Learning + true semantic	**0.9129**	**0.7817**
Connected Components	0.3537	0.2112
Connected Components + true semantic	0.3614	0.2534

4.4 Results

4.4.1 Training details

The volumes are normalized to have unit variance and zero mean. Additionally, most of the air voxels surrounding the specimen are removed by a simple thresholding method. We train and evaluate the network on sub-volumes of $32 \times 32 \times 32$ from the training volume. The sub-volumes are randomly flipped and rotated (by 90, 180 or 270 degrees) during the training phase. For backbones of both the semantic segmentation and embedding learning branch we use the architecture proposed in chapter 3 designed for semantic fiber segmentation (see Figure 3). It is a 3D FCN with standard residual units [47] and batch normalization [58] but with no max-pooling to keep the resolution of the already very thin fibers [70].

The embedding learning is not stable, when trained from noise. Therefore, first we train the semantic segmentation branch for 20,000 iterations and save the weights. Then, we use the weights as an initialization for the embedding learning branch and train it for another 20,000 iterations. The loss which we use for training the embedding learning uses the semantic ground truth masks.

It would also have been an option to share the embeddings and weights for both tasks. Such setup is reported to slightly increase the performance of both semantic and instance segmentation [91]. However, in our setup, we find the above two-stage training to work better. We use 16 feature embedding maps and set α and β to 1 and γ to 0.001. Optimization is done by using the Adam optimizer [64] with a standard initial learning rate set to 0.001. During the prediction phase, the algorithm processes overlapping $32 \times 32 \times 32$ sub-volumes of the test volume with an overlap of 16 in each direction. The post-processing merging algorithm merges the overlapping sub-volumes and produces the final instance segmentation volume.

4.4.2 Results

We compare our method to a standard skeletonization followed by connected component analysis and the Watershed method [136]. For a metric we use the Adjusted Rand Index [55] as proposed in [69] and discussed in the Chapter 2 to measure the performance of instance segmentation. In the method, a binary erosion is first applied on the semantic mask, which serves as seeds after connected component analysis for a watershed segmentation algorithm.

See Fig. 4.6 for a visual comparison. We also evaluate the importance of a good semantic segmentation mask. We provide results for both our method and connected components given the semantic segmentation computed by the semantic segmentation branch as well as using the ground truth semantic segmentation.

Therefore we compare four different setups. Our *Embedding Learning* method using the instance segmentation produced given the semantic segmentation mask from the semantic segmentation branch. *Embedding Learning + true semantic* which is our method but using ground truth semantic segmentation mask instead of the one produced by the methods branch (which is not ideal). *Connected Components* and *Connected Components + true semantic* is the connected component method used either on the output of the semantic segmentation branch or the ground truth semantic mask. We provide results in Table 4.1 for each setup. In the first column, the mean ARI is the mean ARI of all the sub-volumes in the test volume without the merging step. In the second column one can see the score computed over the entire volume after the post-processing merging step which we call a merged ARI. We report the ARI score only for the voxels that belong to the ground truth instance segmentation mask. Including the background voxels would artificially increase the score.

While the standard method clearly fails even when using the true semantic segmentation mask, the proposed method produces meaningful results in all cases. When reasoning on small overlapping patches the proposed method achieves 0.9048 average ARI score. The merging algorithm has trouble with ambiguity of two neighboring outputs and favors merging over splitting. This results in merging two fibers into one, when they are too close to each other. After the merging post-processing step the ARI score decreases to 0.6529.

4.5 Conclusion

In this work, we propose a deep 3D fully convolutional architecture together with a set of post-processing steps for a problem of single fiber segmentation from CT scans of SFRP. We extend a less common approach of embedding learning for the task of 3D instance segmentation. The details of the method together with a post-processing and a merging procedure are explained. The method is significantly superior than the traditional skeletonization - watershed method. We expect our findings to be applicable to a wide variety of volumetric data and not only to fiber composites.

Chapter 5

Deep Learning Based Segmentation Techniques for Medical CT

5.1 Outline

In this chapter we discuss the use of machine learning methods for semantic segmentation for medical CT data. We present a dictionary learning based approach for vessel segmentation from CT thorax scans and a deep learning, U-net based segmentation of liver and liver lesion from lung CT scans together with a boosting training technique named Mask Mining. The work on liver and liver segmentation is based on a joint work with Karsten Roth who has worked under my supervision. The vessel segmentation part is published in the NSS/MIC/RTSD 2016 Conference proceedings [68]. Parts on liver and liver lesion segmentation have been presented at the International Symposium on Biomedical Imaging (ISBI) 2016, International Conference On Medical Image Computing Computer Assisted Intervention (MICCAI) 2017 [113, 8] during the Liver Tumor Segmentation workshops (LITS). Parts on the boosting liver and lesion segmentation by Mask Mining have been presented at the Medical Imaging meets NeurIPS workshop (med-NeurIPS) 2019 and published at the International Symposium on Biomedical Imaging (ISBI) 2020 [112].

5.2 Automated Multiscale 3D Feature Learning for Vessels Segmentation in Thorax CT Images

5.2.1 Introduction

The challenge in vessel segmentation from 3D CT scans is to identify thin vessels at low resolution and the ability to distinguish them from other, similar looking structures. Most of the top-scoring methods in the MICCAI VESSEL12 challenge [114] are based on cost functions derived from the eigenvalues of the Hessian on the image. Such methods usually differ in the selection of scales [33] and interpretation of eigenvalues [36, 73]. Others [132] focus on designing new application-specific kernels.

Feature learning methods via sparse coding are popular for image classification and natural language processing [21]. Instead of using handcrafted filters like in the Hessian-based approaches, the idea is to automatically compute a set of convolution filters particularly tuned to the dataset at hand. The method which has achieved the current highest score in the VESSEL12 challenge [114, 66] uses dictionary learning to obtain convolution filters in an unsupervised manner. All filter responses form a set of feature maps, which are then used to train a supervised classifier for the vessel/non-vessel voxels. Whereas Kiros et al. [66] applies the dictionary learning for 2D patches after designating a preferred slicing orientation, our method extends it to a true 3D approach by using 3D patches. Learning 3D features instead of slice-wise 2D features should increase the accuracy of the classifier at the expense of computation time and memory consumption.

We address the vessel segmentation problem by building upon the multiscale feature learning method of Kiros et al., which achieves the current top score in the VESSEL12 MICCAI challenge. Following their idea of feature learning instead of hand-crafted filters, we have extended the method to learn 3D features. The features are learned in an unsupervised manner in a multi-scale scheme using dictionary learning via least angle regression. The 3D feature kernels are further convolved with the input volumes in order to create feature maps. Those maps are used to train a supervised classifier with the annotated voxels. In order to process the 3D data with a large number of filters a parallel implementation has been developed. The algorithm has been applied on the example scans and annotations provided by the VESSEL12 challenge. We compare our setup with Kiros et al. by running their implementation. Our results show an improvement in accuracy over the slice wise method from 96.66±1.10% to 97.24±0.90%.

5.2.2 Material and Methods

The method is divided into two parts. Dictionary learning for feature learning and a linear classifier for voxel-wise classification of vessel or non-vessel. In order to capture vessels at different scales Gaussian pyramids from the input volumes are generated. We use the multiscale representation during both feature learning (i.e. our patches are sampled from original and scaled volumes at different scales) and classifier learning (i.e. we extract feature maps at many scales).

Dictionary Learning

Feature generation via Dictionary Learning is an unsupervised problem, where from a number of patches the algorithm learns a set of elements that allow for an optimal representation. First we create a set of Gaussian pyramids by convolving input volumes with 3D Gaussian kernels and subsampling them. From the resulting volume data we randomly select a batch of 3D patches $p^{(i)} \in \mathbb{R}^n$ (where n is the number of voxels of the patch) which we use as an input to the sparse coding algorithm for learning a dictionary $D \in \mathbb{R}^{n \times d}$ of d elements, where each column $D^{(j)}$ is one element. We train the dictionary by minimizing the LASSO problem with L1-penalization to ensure the sparsity of the vector $x^{(i)}$ regularized by the parameter λ. That is, we optimize

$$\min_{D,x^{(i)}} \sum_i ||Dx^{(i)} - p^{(i)}||_2^2 + \lambda||x^{(i)}||_1 \tag{5.1}$$

$$\text{subject to } ||D^{(j)}||_2^2 = 1, \forall j$$

over the sparse codes $x^{(i)}$ and the dictionary, D. Fig. 5.1 sketches the dictionary learning step.

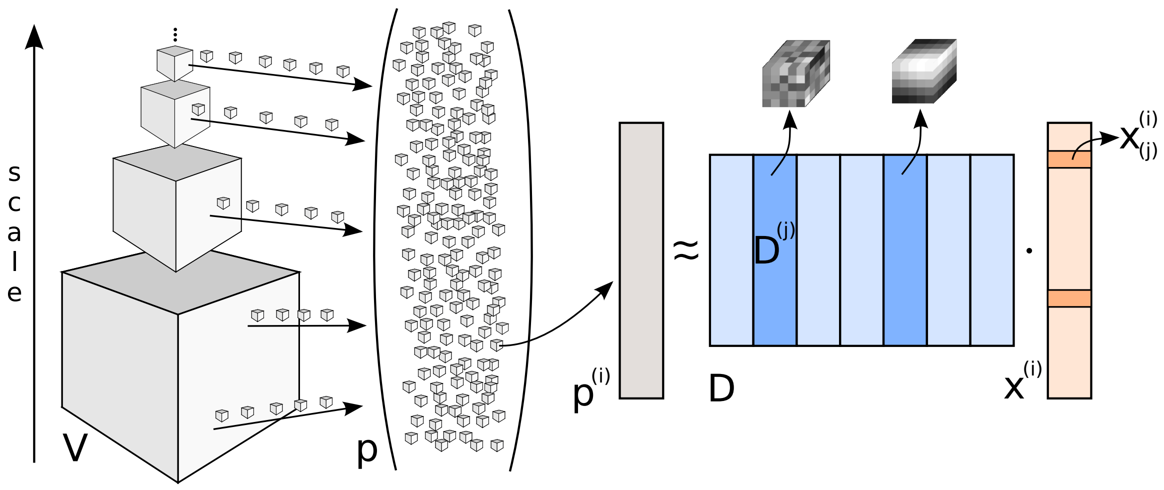


FIGURE 5.1: Dictionary learning overview. From a given volume V a random batch of patches p is extracted. Patches have the same dimensionality as V, but are extracted from different scales of the volume. Dictionary elements $D^{(j)}$ in a dictionary D are learnt in a way to make it possible to reconstruct each patch $p^{(i)}$ by a *sparse* linear combination of the elements. In other words $p^{(i)}$ is approximated by a product of the dictionary and a sparse vector $x^{(i)}$.

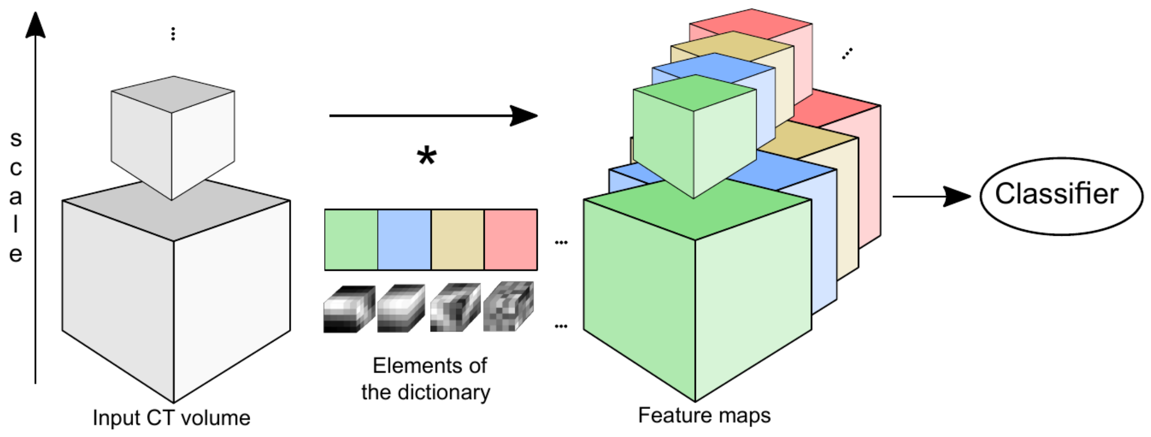


FIGURE 5.2: Feature extraction and classifier learning. The annotated volumes are convolved with the elements of the dictionary producing feature maps at a number of scales. That is one feature volume is created per element and scale. The classifier is learnt on the annotated voxels only. The trained classifier can be evaluated on the entire volume.

Classifier Learning

To classify each voxel as a vessel or non-vessel, we apply supervised learning, making use of manual voxel-wise annotations. As features, we take the convolution response of each element d from the previously learned dictionary D with the annotated volume images at each scale s. Thus, each voxel of a volume contains $s \times d$ predictors. As the number of features is higher than the number of labels in our case, we opt for a linear logistic regression classifier. Fig. 5.2 presents a graphical representation of the step. The hyperparameters of the classifier are tuned using 10-fold cross validation.

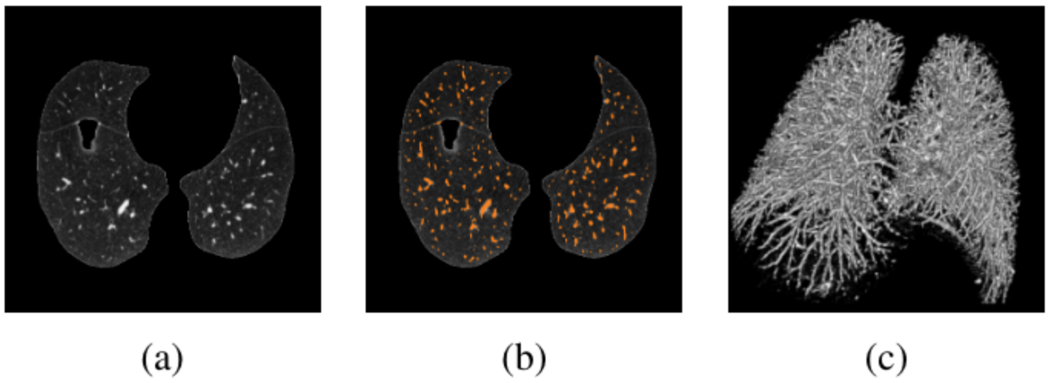


FIGURE 5.3: Result of classification based on learned features. (a) An example slice from the dataset (with lungmask applied), (b) regions classified as vessels by our method (in orange), (c) the 3D visualisation of the regions classified as vessels.

5.2.3 Results

We evaluate our method on the data set provided by VESSEL12 [114]. The data contains 3 annotated volumes with in total 882 annotated voxels (vessel/non-vessel) and additional 20 volumes without annotations. Each volume is of dimension 512×512×512. For each volume, the lung region is denoted by a corresponding mask.

For this work we use a mini batch implementation with least angle regression as a minimizer of the dictionary learning on the 20 given volumes without annotations to train the dictionary D with 512 elements $D^{(j)}$ and $\lambda = 1$. We extract 2,450,000 patches from all available volume images at different scales of the Gaussian Pyramid and normalize them by subtracting the mean. The calculated elements of the dictionary are later used to create feature maps for the 3 annotated volumes from which the 882 annotated voxels are extracted. We divide those voxels at random into a training data set of 657 voxels with which we train the classifier and test data set of 225 voxels with which we evaluate the performance. Annotated voxels are assigned to the training and the test data sets at random. This step is repeated 1000 times in order to average the performance. For the classification learning we achieve best results by using Newton's Method with L2 and number of scales $s = 2$.

Our results for the above setup reached an accuracy of 97.24±0.90% compared to 96.66±1.10% by the slice-wise approach. For the comparison we have used the code and setup provided by Kiros et al. [66]. Fig. 5.3 presents one visual output of our algorithm, and Table 5.1 shows the comparison between the setups.

5.2.4 Conclusion

We show that the 3D approach yields competitive results. We find that using more than 2 scales did not increase the accuracy. One reason may be that the number of features gets too large compared to the number of annotated examples. In the future we plan to gather more training data and use a feature selection method. A significantly higher number of examples should allow us to use more scales and a more flexible model.

Though we use the source code and setup directly provided by Kiros et al., they have reported to use an additional step for the VESSEL12 challenge. Concretely they use an additional layer of depth, to learn additional features from the derived feature maps. This area

TABLE 5.1: Comparison of the results with corresponding parameters.

	Kiros et al.	**Our method**
Algorithm for D training	OMP-1	minibatch LASSO+LAR
Algorithm for logit regression	LBFGS L2	Newton's L2
Dim of patches and elements	5×5	5×5×5
Number of patches $p^{(i)}$	100,000	2,500,000
Number of elements $D^{(j)}$	32	512
Number of scales s	6	2
Number of features	192	1024
Accuracy	**96.66±1.10%**	**97.24±0.90%**

is worth exploring in future research.

5.3 Liver Lesion Segmentation with slice-wise 2D Tiramisu and Tversky loss function

5.3.1 Introduction

The results provided in this chapter are in parts based on the conclusions drawn from the participation at the preceding competition version aimed for ISBI 2017 where the viability of a standard slice-wise *U-Net* architecture was investigated. For MICCAI'17, a pipeline using a *Tiramisu* network and Tversky-coefficient based loss was contributed that achieves very good shape extractions with high detection sensitivity. The goal of the LiTS (Liver Tumor Segmentation Challenge) competition was to compare methods for automatic or semi-automatic segmentation of liver lesions in CT scans. At the present, lesion segmentation is still performed manually (or semi-automatically) by medical experts.

Again, our method is fully automatic. We break down the segmentation into two steps: we first segment the liver (ignoring lesions), then perform a lesion segmentation and liver-mask the final result. This provides a good network-only pipeline that can be improved on by various post-processing methods.

As neural networks proves to achieve human-like performance in biomedical image classification [1], and nearly human-like performance for segmentation tasks [118], it stands to reason to construct a segmentation pipeline that applies these methods for automatic lesion detection.

We evaluate what results can be achieved using the deep fully convolutional neural network architectures *Tiramisu* [62] and the *U-Net* [110] with standard data preprocessing and data augmentation and without any sophisticated postprocessing.

TABLE 5.2: Test results for various pipeline setups. Comparison metrics are Dice (Average/Global), Volume-Overlap Error (VOE), Relative Volume Differene (RVD), Average and Maximum Symmetric Surface Distance (A/MSSD) and Root-Mean-Square-Deviation (RMSD). "DL" denotes the network being trained with dice loss and "TL" with Tversky loss, respectively. "LI" stands for liver segmentation and "LE" for lesion segmentation.

	U-Net LI	U-Net LE	T DL LE	**T TL LE**
DICE(A)	0.95	0.42	0.45	**0.57**
DICE(G)	0.94	-	0.44	**0.66**
VOE	0.10	0.69	0.33	**0.34**
RVD	-0.05	103.74	-0.16	**0.02**
ASSD	1.89	32.54	1.28	**0.95**
MSSD	32.71	-	8.85	**6.81**
RMSD	4.20	-	2.07	**1.60**

The *U-Net* architecture is widely used for biomedical image segmentation, and has already proven its robustness and strength in many challenges (*e.g.* segmentation of neuronal structures [57]). [62] enhanced this structure on the basis of densely connected networks created by [54] to create the *Tiramisu* to achieve better information flow and higher depth.
The complete pipeline resembles the cascaded approach proposed in [100] where the authors suggested a *U-Net* cascade (with an additional subsequent postprocessing step). In our case for liver segmentation, a standard *U-Net* was trained to reduce overall training time, whereas for lesion segmentation, the aforementioned *Tiramisu* was trained from scratch (without transfer learning) with a Tversky loss function to improve on a simple Dice-based loss function.

5.3.2 Material and Methods

The lesion segmentation pipeline includes two networks: a *U-Net* to segment the liver from a given volume slice, and the *Tiramisu* to segment out lesions. By masking the lesion segmentation with the liver mask, we effectively reduce the number of false positives in regions outside the liver. Both architectures will be presented shortly. We implement our networks by use of the deep learning library Keras [29] on the basis of TensorFlow [2].

Dataset and Preprocessing

The dataset consists of 200 CT-scans of the abdominal and upper body region such that the liver is fully included. The data is provided by different clinics from all over the world and is stored in NIfTI dataformat. For working purposes, the set is divided by the LITS challenge organisers into 130 scans for training (containing liver and lesion masks) and 70 for evaluating the method (without masks). From the 130 training scans, we select 25 scans for validation purposes during training and 105 for actual training.

Preprocessing on the provided data includes value-clipping the volume values (on the Hounsfield scale) to an interval of $[-100, 400]$ in order to remove air- and bone-like structures for a more uniform background. The voxel values are then normalized by substracting the training set mean and standardized by dividing through the training set standard deviation. In a final step, the affine matrix is extracted from the NIfTI header to rotate each volume to the same position to make learning easier.

U-Net architecture

Our *U-Net* is implemented with two convolutional layers with padding plus a ReLU-activation layer before max-pooling for the downward movement of the U-branch. These blocks are repeated for a total of four times. We start with 32 filters and a constant 3×3 filter size. The number of filters is doubled after every convolution-max-pooling-block. Dropout is performed after the ReLU-activation layer with a rate of 0.2. The upsampling branch which performs a transposed convolution to reconstruct the segmentation image is implemented in the same fashion. This totals in roughly $7.7 \cdot 10^6$ network parameters.

Tiramisu architecture

The *Tiramisu* is implemented as described in the original paper with four dense blocks comprising four, five, six and seven layers, respectively, in the feature extraction branch and a final bottleneck dense block with eight densely connected layers. Each dense block is created with a growth rate of 12, followed by a maxpooling layer while initializing the network with 32 starting filters. Dropout with probability 0.2 is performed after each dense block, and the extraction structure is mirrored for the upsampling step. ReLU activation is used and L2-regularization with $\lambda = 10^{-5}$ is added to the convolutional layers. In addition, a constant 3×3 filter size is implemented. All in all, this results in approx. $1.8 \cdot 10^6$ parameters, i.e. only a fourth of the standard *U-Net* setup that is used for liver segmentation/ISBI 17. Standard Batch-Normalization is applied before each convolutional layer.

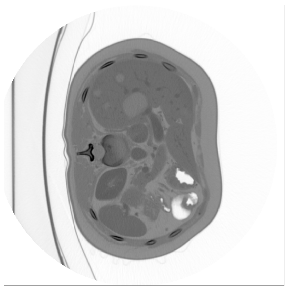
Liver segmentation training

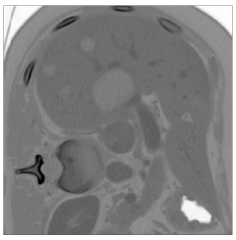
Training is performed over 50 epochs with a batch size of five on a GTX 1070 GPU using an Adam optimizer [64], He uniform initialization [48], standard binary crossentropy loss and a learning rate of 10^{-5} which is halved every 15 epochs. The final weights are selected to provide the best validation score. In total, training on 105 randomly selected volumes (including validating on 25) takes approximately 80 hours. Note that no data augmentation is performed.

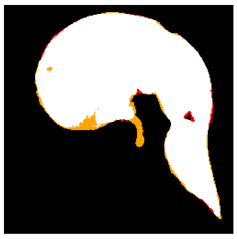
Lesion segmentation training

The *Tiramisu* lesion segmentation network is trained for 35 epochs using the Adam optimization method and the Tversky loss function [115] based on the Tversky coefficient (through simple negation).

For the loss, the coefficient is negated with false positive penalty $\beta = 0.7$ and false negative penalty $\alpha = 0.3$. We penalize false positive predictions higher because training the *Tiramisu* without transfer learning and simple weighted binary cross entropy or dice loss showed the network being prone to a high false positive rate. An initial learning rate of $3 \cdot 10^{-6}$ is


(a) Example slice


(b) Zoomed Liver


(c) Liver Segmentation

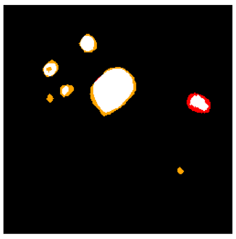

(d) Liver lesion segmentation

FIGURE 5.4: Segmentation of liver and liver lesion from an example slice (a) from the validation data set (volume 46 from the LiTS challenge). The resulting segmentation images (c) and (d) are color coded, where the white color means True Positive, red color False Positive and orange color False Negative.

chosen, which is halved every 10 epochs. The complete training using a batch-size of five required nearly 38 hours, with the usage of very basic data augmentation (minor rotation and translation as well as zooming). During training, the network learns from slices along the coronal axis of which 224×224 crops are taken in and around the liver. In different runs (see Table 5.2), a simple Dice loss is also used for training as reference to the Tversky loss under otherwise same conditions.

5.3.3 Results

After each liver segmentation, only the largest connected component is chosen as the final liver mask. In addition to the test results for the introduced pipeline, Table 5.2 also shows the results for the *U-Net*-only approach for ISBI 17, and a pipeline with a *Tiramisu* lesion segmentation network that is trained with a simple dice loss function. Each method is applied on the 70 test volumes and shares the same liver segmentation network. Total inference time including both liver and lesion segmentation takes roughly seven seconds per volume on average, totaling in 8 hour of testing time on a GTX 1070 GPU. For a measurement of tumorburden, the pipeline scores 0.03 on RSME and 0.18 on Max Error.

5.3.4 Conclusions

The major strength of this approach is its simplicity, as we use a network-only based segmentation approach to achieve good segmentation results (see Table 5.2 with comparison to our *U-Net*-only approach for ISBI 17).

Moreover, although placed in the mid-ranks Dice-wise, the proposed pipeline achieves best scores regarding Volume-Overlap Error (VOE) and Average Symmetric Surface Distance (ASSD) as well as scoring high in Relative Volume Difference (RVD) and Maximum Symmetric Surface Distance (MSSD). This insinuates that the Tiramisu performs very well when it comes to actual shape extraction when lesions are detected correctly.

However, there are several issues solving which could improve the segmentation capabilities:

(1) As can be seen from Fig. 5.4 and Table 5.2, our liver segmentation performs reasonably well. However, the method misses a majority of very big and/or lesion at the liver boundary which then, although maybe detected through the lesion segmentation network, get masked away. Increasing the training effort for our liver segmentation network would therefore allow us to better our current lesion (and liver) segmentation results.
(2) Even with our current implementation of the Tversky loss function, the major error source are false positive prediction. Higher penalties, a network ensemble and regularization through, for example, higher dropout or more sophisticated data augmentation would ideally increase the area under the ROC curve and provide better overall scores.

In addition, training a liver segmentation network on the basis of a *Tiramisu* and using that as weight initialization could potentially help the network to converge faster and to better results.

On a final note, [100] and [18] shows that good post-processing (for example with 3D conditional random fields and random forests with additional handmade features) could improve the result. We feel this is an avenue worth exploring on this data. Also, segmentation is only performed on a 2D slice-by-slice basis. [44] shows exceptional results through the application of a semi-3D approach by including additional upper and/or lower slices in the segmentation process of the neural network to incorporate more spatial information. Our pipeline can be easily extended to make use of this approach to hopefully improve the results further.

5.4 Boosting Liver and Lesion Segmentation from CT Scans by Mask Mining

In this work we propose a novel procedure to improve liver and liver lesion segmentation from CT scans for U-Net based models. Our method is an extension to standard segmentation pipelines allowing for more fine-grained control over the network output by focusing on higher target recall or reduction of noisy false-positive predictions, thereby also boosting overall segmentation performance. To achieve this, we include segmentation errors into a new learning process appended to the main training setup, allowing the model to find features which explain away previous errors. We evaluate this on semantically distinct architectures including cascaded two- and three-dimensional as well as combined learning setups for multitask segmentation. Liver and lesion segmentation data is provided by the Liver Tumor Segmentation challenge (LiTS), with an increase in dice score of up to 2 points.

5.4.1 Introduction

As the liver is an essential detoxification organ, regular control and evaluation of health or disease progression is crucial, especially since primary tumors from breast, colon or pancreas often metastate into the liver [8]. Therefore, early evaluation and staging is essential for preventive measures, with Computed Tomography (CT) being most commonly used for imaging [8]. Providing fully-automatic segmentation of liver and liver lesion tissue from CT data can hence be a useful tool to help with diagnosis and treatment planning. As Encoder-decoder style neural networks, in particular U-Nets [110], have given rise to fully-automatic state-of-the-art solutions for many medical segmentation tasks, e.g. [59, 8, 26], it stands to reason to utilize those. However, high dimensionality and non-linearity make the training of neural networks a difficult endeavour. To reach metric baselines or improve on existing scores requires computationally expensive re-runs without guarantee of improvement. In addition, especially in medical applications, neural networks have to comply with expectations on sensitivity or robustness towards false-positive predictions. To get both reliable improvement and control in performance, the inclusion of segmentation errors into the training procedure therefore seems logical. We thus propose a novel pipeline to reliably boost network segmentation performances and allow for stronger control over the final network segmentation behaviour. Our approach separates the standard network training procedure from the inclusion and control of segmentation errors. In doing so, we stay independent of architecture and data choices. Using segmentation error types and specific loss functions as new training signals, we are able to offer a framework helping networks explain away own segmentation errors, thereby boosting segmentation performance and allowing for easy control of network outputs.

Related work

Our strategy differs from prior work on the inclusion of segmentation errors into the training process include [115, 113], who propose a Tversky-coefficient-based loss, which generalizes the standard Dice coefficient loss to include additional hyperparameters for penalizing false-positive or false-negative predictions during training. [107] utilize segmentation error types in a complex adversarial setup, where refinement networks are trained on top of the basic setup to remove these errors. While the former introduce new hyperparameters, the usage of adversarial networks in [107] limits the usable network architectures. In both cases, heavy tuning and reruns are required for different architectural setups, as these methods are linked directly to the main learning process. This holds especially true going to three-dimensional data, which is common for many medical segmentation tasks. This limits the controllability of the final network performance and, if applicable, requires heavy tuning for different architectural setups, especially going to three-dimensional data, which is common for many medical segmentation tasks.

We therefore propose to use segmentation error types in a setup separate to the main training. Using segmentation errors of the learned networks, we append a secondary training process with specific loss functions to provide a framework that helps networks explain away own segmentation errors, thereby boosting segmentation performance for qualitative impressions). This means that our method stays independent of architecture and data choices, and allows for improved performance without costly reruns of the full setup.

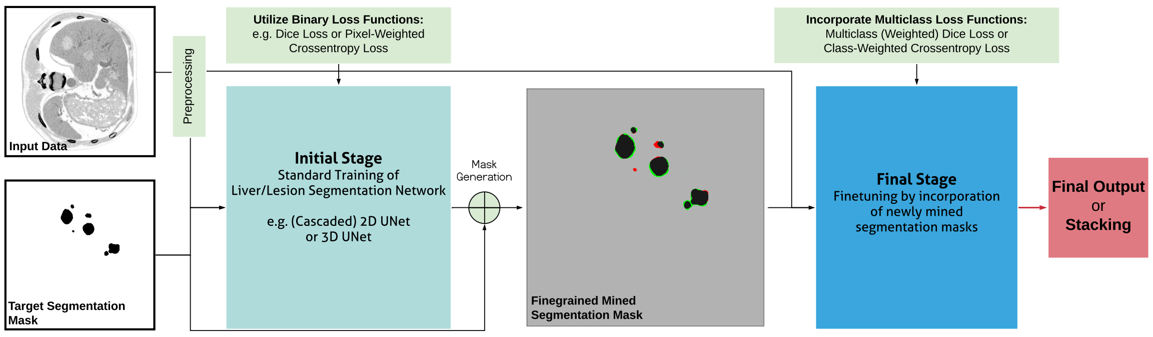

Input Data
Target Segmentation Mask
Preprocessing
Utilize Binary Loss Functions: e.g. Dice Loss or Pixel-Weighted Crossentropy Loss
Initial Stage
Standard Training of Liver/Lesion Segmentation Network
e.g. (Cascaded) 2D UNet or 3D UNet
Mask Generation
Finegrained Mined Segmentation Mask
Incorporate Multiclass Loss Functions: Multiclass (Weighted) Dice Loss or Class-Weighted Crossentropy Loss
Final Stage
Finetuning by incorporation of newly mined segmentation masks
Final Output or Stacking

FIGURE 5.5: The Mask Mining Pipeline. Starting from the original training setup, generated segmentation masks are compared with the ground truth masks to generate new finegrained multiclass training masks containing previously made segmentation errors. This allows the network to learn to explain away mistakes.

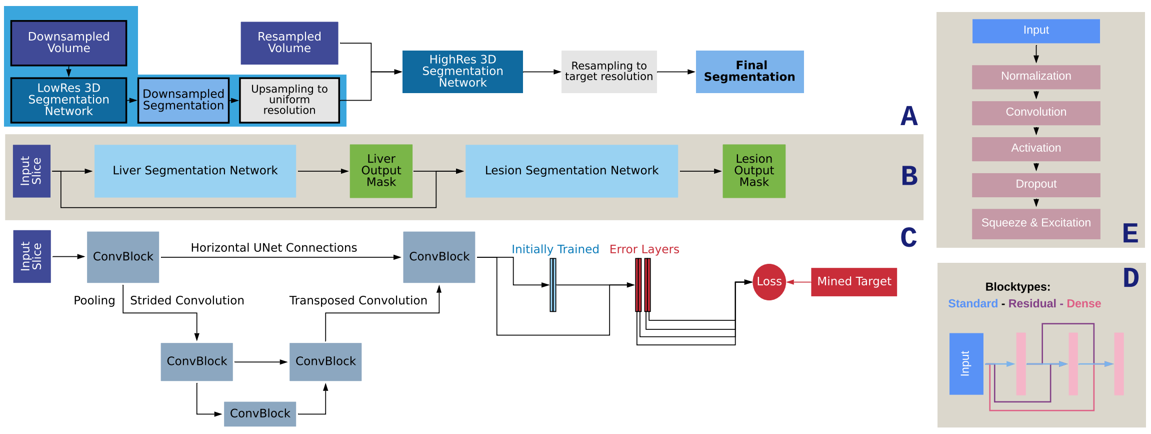


FIGURE 5.6: Schematics of all utilised architectures and training pipelines. **(A)** (Optional two-step) pipeline for 3D liver/lesion segmentation networks. **(B)** Setup for simultaneous training of liver and lesion segmentation networks. **(C)** Basic U-Net architecture, including the additional kernelsize 1 error layers to train on error masks. **(D)** Convolutional U-Net Block with either dense, residual or basic connectivity. **(E)** Layers utilized to represent a single ConvBlock-Layer (pink in (D)).

Distinctly different networks and datatypes are tested to check the architecture- and datatype-independent applicability. This includes 2D and 3D data utilised in different training styles which are based around 2D and 3D U-Net [110, 20, 126] pipelines (Fig. 5.6). Both training and evaluation is done on the Liver Tumor Segmentation (LiTS) dataset [8], showing consistent improvements in all setups. Note that throughout this chapter, we will refer to the usage of our pipeline as *Mask Mining*.

5.4.2 Material and Methods

Fundamental for our proposed extension (Fig. 5.5) is the generation of new training masks to alter the current network performance and allow the network to learn from its own errors.

A segmentation pipeline of choice is trained until convergence following any training procedure. Now, segmentation masks over the training data are generated through single forward passes with minimal computational burden. These are then compared to the original ground truth to determine new training classes for each pixel, based on segmentation error cases: *True Negative*, *False Positive*, *False Negative* and *True Positive*. This gives four target classes compared to the binary case with two classes. We then append four single-layer output channels serving as error prediction layers to the output layer, introducing no relevant new parameters, but ensuring that all previously learned weights are kept until retraining on the novel masks is performed (see Fig. 5.6C). Due to the initial pretraining, convergence occurs much faster. The extended training separates into two different setups, focusing around utilised loss functions for different attributes.

Assuming the majority of predicted pixels to be true positive or negative after training, we distinguish two approaches based on the choice of loss:
A pixel-weighted crossentropy loss (pwce) (e.g. [110]) gives highest learning signal to high frequency targets. As we have a high imbalance towards true positive/negative predictions, retraining on error masks primarily reinforces these predictions while dropping noisy false positives. The retrained multiclass error case predictions are then grouped into true positive/false negative and true negative/false positive predictions to generate a final binary segmentation mask. To evaluate the network on the standard segmentation task, we find the argument maximum of the new multiclass predictor ϕ^{multi} for image x^k and compute the respective binary segmentation mask O^k via

$$O^k_{ijm}(x^k) = \lfloor \frac{\text{argmax}_{c \in [0,..,C-1]}\ \phi^{multi}_{ijm,c}(x^k)}{2} \rfloor \tag{5.2}$$

for i, j, m iterating over height, width and depth (zero for 2D data). Using this approach, we merge true positive and false negative prediction to foreground and false positive and true negative pixels to background.
A dice-coefficient based loss (e.g. [26]) injects a stronger learning signal for underrepresented classes, motivating higher recovery of false-negative and false-positive pixels. As the primary interest now lies in explaining away obfuscating features while retaining crucial ones, we replace the True Positive class with the original segmentation mask. This allows the network to transfer properties generating false-positive segmentations to the respective output channel and recover generators for false-negative predictions for higher target pixel recall and boosted performance. The binary segmentation masks are directly extracted from the former True Positive output channel.

Data and Pre/-Postprocessing

The Liver Tumor Segmentation (LiTS) dataset [8] contains 131 three dimensional CT scans of the lower abdominal area with ground truth masks for liver and liver lesion tissue, as well as 70 test volumes. Those are evaluated by online submission to the dataset webpage. All volumes have horizontal dimensions of 512 with near constant resolution. In the axial direction, dimensionality and resolution varies strongly, which is a relevant factor for any approach requiring higher-than-two dimensional data input. Before training, the data is bounded to $[-100, 600]$ HU before performing standardization with dataset mean and standard deviation. For evaluation, only the largest connected component is used to generate the final liver segmentation, after applying minor binary erosion and dilation to remove tiny extrusions.

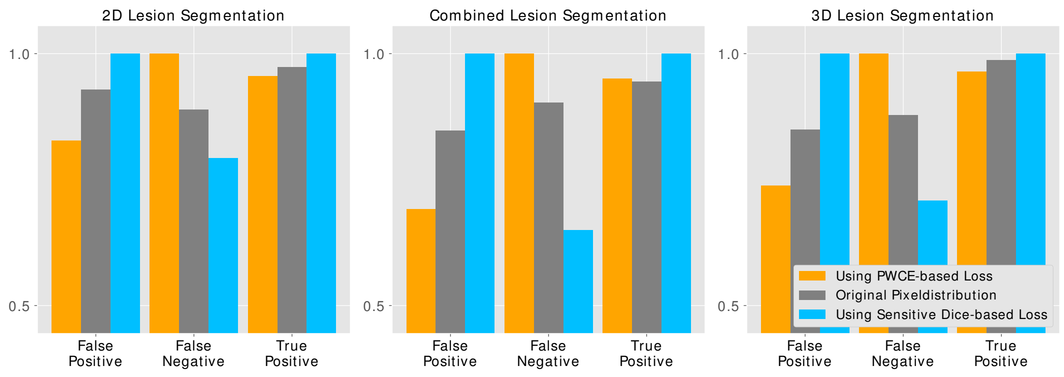


FIGURE 5.7: Qualitative examination of the control capability of our pipeline extension. We evaluate false positive/false negative/true positive pixel count change for a fixed validation set on lesion segmentation capability for all architectures (sec. 5.4.3). Each bar group is normalized to the highest group value for visual clarity. We see that the network performance can be tweaked retrospectively for higher recall (blue) or robustness (orange) without adding new hyperparameters and independent of the underlying architecture.

5.4.3 Results

We investigate the performance of our pipeline on liver and lesion segmentation by evaluating dice score performance on distinct architectures described in the previous section and the control capability of our extension. Each pipeline is trained to convergence before applying our extension to ensure that we do not just prolong the training process.

Evaluated Architectures & Loss Functions

For representative 2D and 3D pipelines, we use our own implementations following [19] to solve the multiclass segmentation problem of liver and liver lesion segmentation using a cascaded training approach. The underlying network architecture utilizes standard 2D and 3D U-Nets [110, 20] with residual blocks [47]. Squeeze-and-Excitation modules [53] and multislice inputs following [44] are added as well. In the 2D case, standard Batch Normalization [58] is used. To accommodate for the high variance in axial resolution and high memory requirements in the 3D approach, we utilize a two-step approach following [59] by first training on down- and then resampled data. The latter is extended with upsampled segmentations on the downsampled data before training. Finally, similar to [126], a combined training of the network cascade is included as well. Using two smaller U-Nets, both are trained simultaneously on liver and liver lesion segmentation respectively. The lesion segmentation network receives as input the predicted liver segmentation mask as well as the original input. Schematics are visualized in Fig. 5.6 for all three pipelines.

Initial training is done using pixel-weighted categorical crossentropy (pwce) loss with distance-transformation weightmaps (see [110]) for liver and a loss based on pwce, L^{pwce}, and a smooth dice score, marked as L^{dice} (see e.g. [26]), for lesion segmentation:

$$L_{combined}(x^k, t^k, w^k, \phi) = L^{pwce}(x^k, t^k, w^k, \phi) \cdot \left(L^{dice}(x^k, t^k, \phi) + \epsilon\right)^{-1} \tag{5.3}$$

with images $\{x^k\}_{k\in[1,K]}$ in minibatch of size K, network ϕ, target mask t^k and weightmap w^k with width W and height H. For numerical stability, a small $\epsilon = 10^{-5}$ is added.

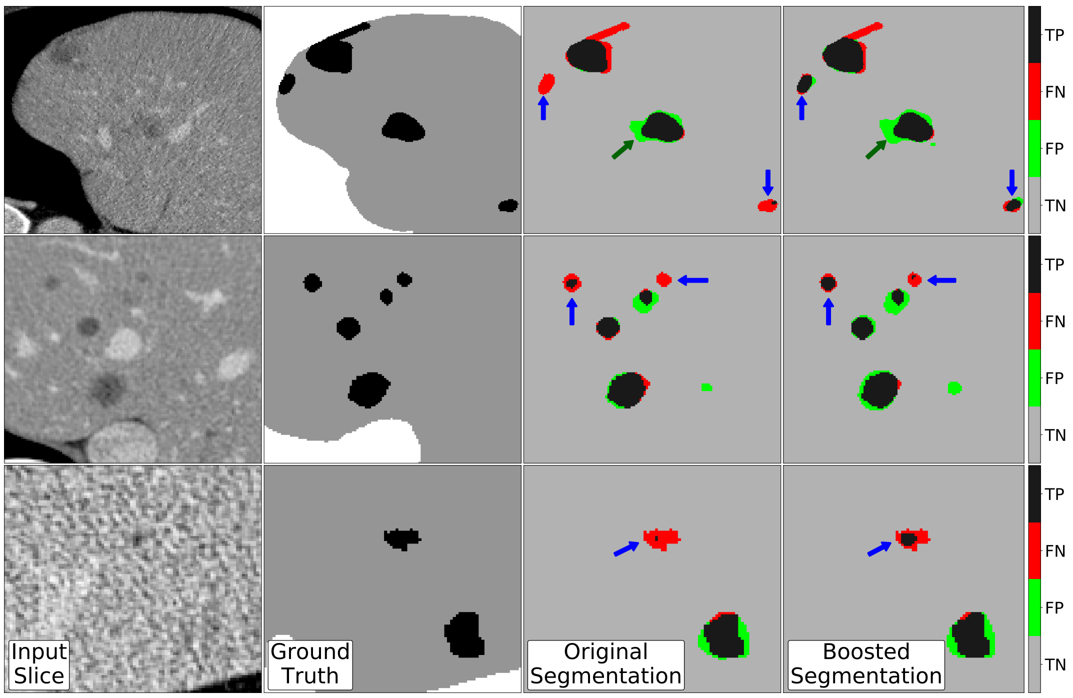


FIGURE 5.8: Qualitative evaluation of our pipeline extension. Dice-based loss function on mask mined training. *Original Segmentation* denotes the segmentation errors for the initial network, while *Boosted Segmentation* shows the performance after one iteration of mask mining. It can be seen that the number of false negatives (*red*) is clearly reduced, with a slight introduction of new additional false positive segmentations (*green*). *Grey* and *black* denote true negative and positive errors.

Implementation Details

All architectures as well as the overall mining pipeline are implemented using the PyTorch framework [99]. The training data is divided in a 85%|15% split. We run everything on a single NVIDIA GeForce 1080Ti. 288×288 crops with batchsize 8 are used for 2D training and $128 \times 128 \times 64$ crops with batchsize 2 3D training. For liver segmentation, crops are taken randomly, while for lesion segmentation crops in and around the liver are used. Standard data augmentation using random horizontal and vertical flips, random rotation and random zooming is performed, all in axial direction. For optimization, Adam [64] with an initial learning rate of 10^{-5} and $L2$-regularisation $\lambda = 10^{-5}$ is used. Standard step-based learning rate scheduling is included as well. Training is performed for 70 epochs to ensure convergence, saving the best validation weights.

Control Study

To examine the control capabilities, we note the initial distribution of segmentation error types for a withheld validation set (same reasons as previously), before running a mask mining step with a multiclass dice loss as well as a multiclass pwce loss. This is done for all three architectures. Summarized in Fig. 5.7, we see a clear shift in false-positive and false-negative pixels depending on the choice of utilized loss. Note that the standard loss setup is used without any parameter tuning. As can be clearly marked out, the network segmentation behaviour drastically changes for different loss functions, while boosting the segmentation performance in both cases. This is done only with the computed error training masks without including any external input.

TABLE 5.3: *Quantitative evaluation of network performance before and after application of mask mining.* We show volume-averaged dice scores for liver and lesion segmentation on the test set and a fixed training and validation set. The evaluated architectures are described in Sec. 5.4.3. We see a clear improvement in segmentation dice scores, especially for the combined approach due to the simultaneous inclusion of liver and liver lesion corrections. In addition, error inclusion reduces seed-dependent variation in performance (measured over three runs).

Setup	Training Dice		Validation Dice		Online Test Dice	
	Liver	Lesion	Liver	Lesion	Liver	Lesion
2D Cascade	96.9 ± 0.3	71.9 ± 0.4	95.9 ± 0.3	63.5 ± 0.6	95.3 ± 0.2	62.9 ± 0.3
Mask Mined	$\mathbf{97.0 \pm 0.1}$	$\mathbf{73.7 \pm 0.2}$	$\mathbf{96.3 \pm 0.2}$	$\mathbf{64.9 \pm 0.2}$	$\mathbf{95.5 \pm 0.3}$	$\mathbf{63.5 \pm 0.2}$
3D Cascade	92.2 ± 1.4	63.0 ± 0.8	91.4 ± 0.9	56.8 ± 2.0	91.2 ± 1.0	55.5 ± 0.9
Mask Mined	$\mathbf{94.2 \pm 0.3}$	$\mathbf{66.1 \pm 0.4}$	$\mathbf{91.8 \pm 0.6}$	$\mathbf{57.7 \pm 0.4}$	$\mathbf{92.0 \pm 0.4}$	$\mathbf{56.5 \pm 0.2}$
Combined	94.5 ± 0.3	70.1 ± 0.5	92.9 ± 0.7	61.6 ± 0.5	93.4 ± 0.3	61.9 ± 0.2
Mask Mined	$\mathbf{96.2 \pm 0.5}$	$\mathbf{72.3 \pm 0.4}$	$\mathbf{94.0 \pm 0.3}$	$\mathbf{63.4 \pm 0.4}$	$\mathbf{94.7 \pm 0.3}$	$\mathbf{63.0 \pm 0.1}$

Training, Validation and Test Performances

We compute the averaged dice score per volume on the test volumes before and after mask mining. This is done on three semantically different architecture types (3D and 2D cascaded as well as combined 2D training) to examine the general applicability of our pipeline extension. Hence, relative improvement is the key metric to examine.

In detail, due to biggest dice score improvements in initial testings, a dice-based multiclass loss (Equation 5.3) setup is utilized (see Fig. 5.6). Results are summarized in Table 5.3, showing a consistent gain over the initially trained model, especially for the combined training setup. This is arguably due to the simultaneous boost in liver and lesion segmentation performance.

The inclusions of mined trained masks into the training process specifically benefits validation performance. This is rooted in the splitting procedure, as training and validation set are drawn from the same sample set. Due to different sources contributing to the dataset [8], the test set samples therefore differ much stronger from the training set. Newly mined features are therefore more expressive on the validation set.

Additionally, qualitative impressions pre/post mask mining are shown in Fig. 5.8 using the same setup. We see a higher lesion recall with a small increase in false positive predictions.

5.5 Conclusion

We introduced a novel extension to standard liver and lesion segmentation pipelines on the basis of the Liver Tumor Segmentation (LiTS) dataset. By helping the network learn and thereby explain away previously made errors using automatically generated training labels, we boost segmentation performance on different and distinct architectures and training styles.

Although we present our work on the task of liver and liver lesion segmentation from CT scans via deep learning U-net like architectures, due to the architecture-independent applicability our method can be extend to other medical image segmentation problems. Not only to a variety of other applications but also to other machine learning based semantic segmentation techniques. We see our Mask Mining idea as an addition to boosting and ensemble methods.

By separating the boosting step from the basic training setup we allow for straightforward post-training correction of network segmentation performance and the inclusion of meaningful segmentation attributes like sensitivity and robustness towards false positive predictions. Decreasing the number of false negatives is of great significance, especially in medical image analysis. Most significantly, the use of our Mask Mining method allows potential detection of previously omitted objects of interest.

However, we still observe limitation in terms of the dice score. A straightforward idea for improvement would be an iterative approach in which the Mask Mining could be repetitively used on the learning model. It should help the model to further increase the sensitiveness to the errors performed by the model. We leave this issue for future research and investigation.

Chapter 6

Plugin Networks for Inference under Partial Evidence

6.1 Outline

In this chapter we propose a novel method to incorporate partial evidence in the inference of deep convolutional neural networks. Contrary to the existing, top performing methods, which either iteratively modify the input of the network or exploit external label taxonomy to take the partial evidence into account, we add separate network modules ("Plugin Networks") to the intermediate layers of a pre-trained convolutional network. The goal of these modules is to incorporate additional signal, i.e. information about known labels, into the inference procedure and adjust the predicted output accordingly. Since the attached plugins have a simple structure, consisting of only fully connected layers, we drastically reduced the computational cost of training and inference. At the same time, the proposed architecture allows to propagate information about known labels directly to the intermediate layers to improve the final representation. Extensive evaluation of the proposed method confirms that our Plugin Networks outperform the state-of-the-art in a variety of tasks, including scene categorization, multi-label image annotation and semantic segmentation. Work from this chapter is published in the Winter Conference On Applications of Computer Vision (WACV) 2020 proceedings [72].

6.2 Introduction

Visual recognition tasks, e.g. scene categorization or multi-label image annotation, have attracted a significant amount of research interest in recent years [128, 52, 133, 30]. One of the reasons, which sparks this attention is the availability of evaluation datasets created for benchmarking given visual tasks, such as ImageNet [23], VOC Pascal [28] or COCO [79]. Although sensible for comparison purposes, single-task evaluation protocols are often far from real-life use-cases, where additional information, e.g. related to location or time of photo capture, is available.

The availability of partial information (*partial evidence*) about an image, made available at test time, can improve accuracy of pre-trained networks [52, 128], and we will follow this scenario. More specifically, we assume that a set of labels corresponding to a given image is known during inference, while the task at hand is to improve the performance of the

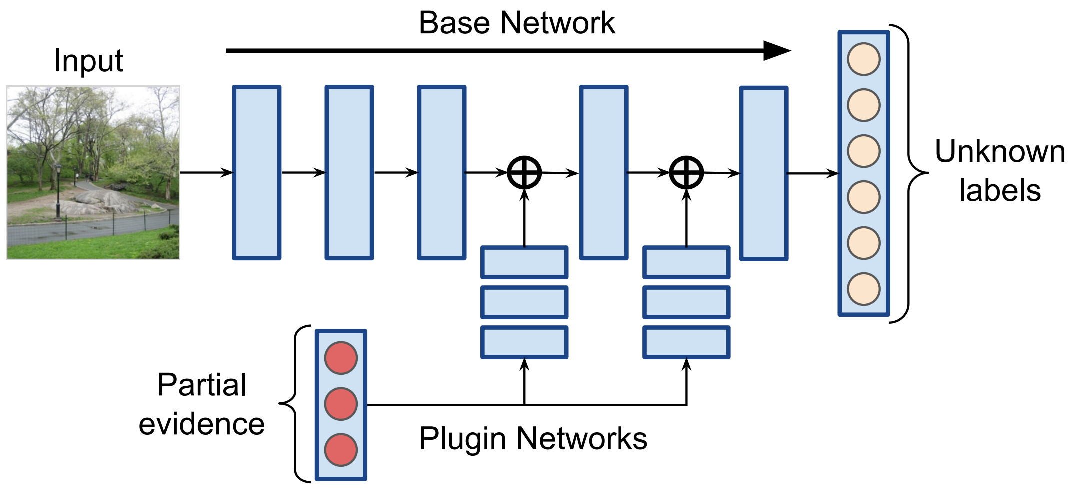


FIGURE 6.1: Plugin Networks — neural networks attached to the intermediate layers of a pre-trained convolutional neural network, allow to exploit partial evidence labels at the inference to predict unknown labels with higher accuracy. This simple, yet effective approach significantly reduces train and test time with respect to the state-of-the-art, while outperforming competitive results on three challenging benchmarks.

model on the original task, e.g. image classification, object detection, semantic segmentation. This corresponds to a real life application, where, for instance, we know that the image was captured in a forest or in a cave, which drastically reduces the likelihood of detecting a skyscraper. Similarly, information that a given object appears in an image can greatly improve its localization or segmentation. Since partial evidence can be available in multiple forms and modalities, the main prediction system, e.g. a convolutional neural network (CNN), is trained to perform a general purpose prediction with no assumption about the existence of partial evidence or lack thereof. Neural architectures such as CNNs are not modular, thus any modification such as new inputs (partial evidence) or new outputs (new tasks) are difficult to apply without repeating the full training procedure. Otherwise, phenomena such as the catastrophic forgetting may occur. Our objective is to enable the model to incorporate additional available information without re-training the main system while exploiting this information to increase the quality of predictions.

Several methods are proposed in the literature to address this problem, among them [52] and [128] are the most recent. In paper [52] authors propose to exploit external taxonomy of the labels by modelling correspondences between scene attributes and categories, by feeding this data into the main neural network at inference. On the other hand, paper [128] introduces a feedback-prop approach that iteratively modifies input and network activations to ensure the response of the network corresponds to the distribution of known labels. Although, those methods provide effective ways to exploit partial evidence, they require complex reasoning, that concerns relationships between labels or computationally expensive iterative adaptation mechanism.

In this work, we reduce the complexity and propose the Plugin Networks – a simple, yet effective main network extension that allows to incorporate partial evidence during the inference. We show that by using a set of fully-connected (FC) side networks attached to intermediate layers of the main network (see Fig. 6.1), we are able to not only avoid costly optimization process, but also exploit the assumption about the existence of partial evidence in the offline training stage. More specifically, the proposed Plugin Networks, connected to the backbone neural network, adjust their activations at the time of inference, depending on available known labels. Due to the simplicity of the Plugin Networks, their training converges quickly, while remaining robust to overfitting, as we show in this work. The inference of the proposed model consists of a quick feed-forward propagation of the main model. Plugin Networks offer a significant speedup with respect to the state-of-the-art feedback-prop method [128]. Last but not least, the proposed Plugin Networks outperform all of the existing methods on three challenging benchmark applications: hierarchical scene categorization on the SUN397 dataset [133], multi-label image annotation on the COCO 2014 dataset [79], and semantic segmentation on Pascal VOC 2011 [28].

To summarize, the contributions of our work are the following:

- We propose novel neural network model extensions called Plugin Networks, which allows us to take partial information available at test time into account. Plugin Networks adjust the activations of the pre-trained base network. They are fast to train and efficient at test time.
- We show how to attach the proposed Plugin Networks to different types of neural network layers and investigate the influence of those variants on final results.
- We provide an extensive evaluation of the proposed approach on three challenging tasks: hierarchical scene categorization, multi-label image annotation and scene segmentation.

In the remainder of this work, we first give an overview of related publications. In Sec. 6.3 we formally introduce the proposed approach, explain how to use it and discuss its properties. Sec. 6.4 provides an extensive evaluation of our method and we conclude this work in Sec. 6.5.

6.2.1 Related Work

Using context in visual tasks: Exploiting additional contextual cues in visual recognition tasks gained a lot of attention from the computer vision community [25, 40, 61]. Contextual information related to semantics is used to improve object detection [90]. Social media meta-data is also used in a context of multilabel image annotation in [61]. Although, adding context proved to be successful in increasing the quality of visual recognition tasks, all of the above mentioned methods used the context in conjunction with the input uni-modal (visual) image during the training of the entire system. In this work, we propose a fundamentally different approach since the context (in the form of known labels) is learned only after the training of the main model is finished and our approach allows to extend this pre-trained model with additional information *a posteriori*. Rosenfeld *et al.* [111] proposed a method where detection and segmentation is conditioned on the presence of a given object category. To achieve it, they propose to use a set of linear modulators. This method shares common features such as offline training with the Plugin Networks, but a model capacity of linear modulators is not enough to learn complex functions. This leads to more complicated training procedures with data oversampling.

Using label structure: Some authors propose to model the co-occurrence of labels available at training time to improve recognition performance [89]. [24] on the other hand uses a special structure to store the relations between the labels using a graph designed specifically to capture semantic similarities between the labels. Other forms of external knowledge can be found in [42] and [56] where they use the WordNet taxonomy of tags to increase the accuracy of their visual recognition systems. [61], [87] also use social media meta-data to improve the quality of the results obtained for image recognition tasks. Finally, [95] estimates entry-level labels of visual objects by exploiting image captions. Contrary to our method, the above-mentioned approaches focus on finding the relationships between the labels and driving the training algorithm to encompass those structures. In this work, we do not explicitly model any label structures – the only input related to labels we give to the network is a set of known labels related to an image with no information about their relationship with the others.

Multi-task learning: Somehow related to our work is the thriving area of multi-task learning. Motivated by the phenomenon of catastrophic forgetting, multi-task learning tries to address the problem of lifelong learning and adaptation of a neural network to a set of changing tasks while preserving the network's structure. In [77], Lee *et al*. aim to solve this problem by the continuous matching of network distribution. In [105] the same problem is solved through residual adapters – neural network modules plugged into a network, similarly to our Plugin Networks – which are the only structures trained for the tasks while the base network remains untouched. Although, we do not aim to solve multi-task learning problem in this work, our approach is inspired by the above-mentioned methods, which focus on designing robust network architecture that can dynamically adjust to additional data point sources unseen during training.

Inference with Partial Evidence: Finally, the most relevant to the work presented in this chapter are two methods proposed by Hu *et al*. [52] and Wang *et al*. [128]. Both of them address the problem of visual tasks in the presence of partial evidence.

Hu *et al*. [52] tackles this challenge by proposing a Structured Inference Neural Network (SINN). The SINN method is designed to discover the hierarchical structure of labels, but it can also be used in a partial evidence setup, if labels in a given hierarchy are clamped at inference. However, the SINN model, which uses CNN and LSTM to discover label relations, has a large amount of learnable parameters, which makes model training difficult. To solve this issue authors use the positive and negative correlations of labels as prior knowledge, which is inferred from the WordNet relations. We compare our method with SINN and show that we achieve significantly better performance with a much simpler model.

The FeedbackProp proposed by Wang *et al*. [128], uses an iterative procedure, which is applied at inference time. The idea is to modify network activations to maximize the probabilities of labels under the partial evidence. The method does not require to re-train the base model. However, due to the iterative procedure introduced at inference time, it requires more computational effort. In addition, they introduced hyperparameters, like a number of iterations and learning rate to the inference phase. Finally, in the case of FeedbackProp, the partial evidence labels can only be a subset of labels that the base model can recognize. Our method, however, can accept any kind of labels as partial evidence. Moreover, our method introduce negligible computations, and not extra parameters to the inference phase. The comparison shows that our method outperforms FeedbackProp while being significantly faster at inference phase.

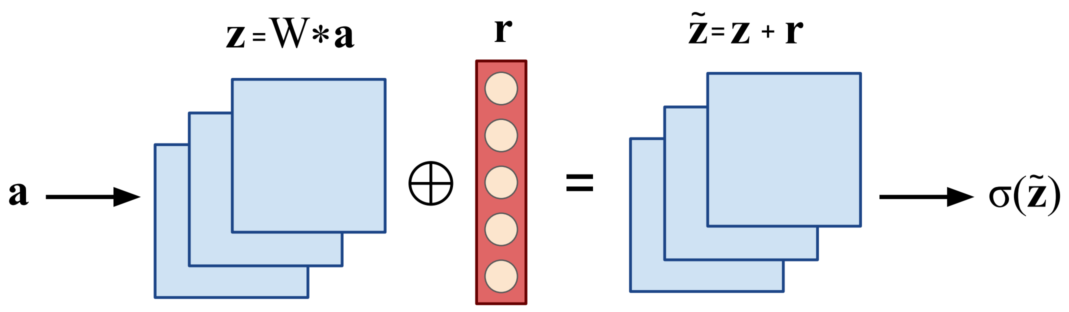


FIGURE 6.2: If we add Plugin Network to a convolutional layer, then each element of the output vector is added to a corresponding channel of feature maps. For instance, if convolutional layer has c channels, then output from the Plugin Network has also c elements.

6.3 Material and methods

In this section, we first introduce the Plugin Networks and define them formally. We then describe how to attach the Plugin Networks to the existing base network at the linear and convolutional layers.

6.3.1 Definition

Let's assume that we have a CNN model $F(\mathbf{x};\mathbf{w})$, where $\mathbf{x}$ is an input image and $\mathbf{w}$ are the parameters. The model F is already trained on some task (*e.g.* single or multi-label classification, scene segmentation). The parameters $\mathbf{w}$ were trained on input images X and input labels Y.

Now let's assume that some labels $\bar{Y}$ are available and known at inference time. In the following definitions without loosing generality, we will assume that only one Plugin Network is attached to the base model. We define the Plugin Network model F_p with parameters $\mathbf{w}_p$ as

$$\mathbf{r} = F_p(\bar{\mathbf{y}};\mathbf{w}_p). \tag{6.1}$$

The model takes the partial evidence $\bar{\mathbf{y}} \in \bar{Y}$ as an input. The output $\mathbf{r}$ of the plugin can be attached to the output vector $\mathbf{z}$ of some layer of the base model F:

$$\tilde{\mathbf{z}} = \mathbf{z} \oplus \mathbf{r}, \tag{6.2}$$

where the sign $\oplus$ can have the following meaning:

- additive: $\tilde{\mathbf{z}} = \mathbf{z} + \mathbf{r}$,
- multiplicative: $\tilde{\mathbf{z}} = \mathbf{z} * \mathbf{r}$,
- residual: $\tilde{\mathbf{z}} = \mathbf{z} + \mathbf{z} * \mathbf{r}$.

In this way the Plugin Network F_p adapts the output vector $\mathbf{z}$ of the base model F under presence of available partial evidence. The eq. (6.2) defines how the Plugin Network F_p is attached to the base network F, thus a joint model can be defined as:

$$\tilde{F}(\mathbf{x},\bar{\mathbf{y}};\mathbf{w},\mathbf{w}_{p_i}). \tag{6.3}$$

In general, several Plugin Networks can be attached simultaneously to a number of layers of the base model F.

Note that the output of a Plugin Network $\mathbf{r}$ can be attached to either the output of a fully connected layer or a convolutional layer. In the following sections, we explain how both operations are performed in details.

6.3.2 Connection with Linear Layers

The Plugin Network, which is attached to the linear layer, has to compute the vector $\mathbf{r}$ of the same dimension as the vector $\mathbf{z}$. Then the operator $\oplus$ in eq. (6.2) is well defined. The adjusted vector $\tilde{\mathbf{z}}$ is then processed by the following layers of the base model. The value $\mathbf{z}$ is the output of a layer before a non-linear function (*e.g.* ReLU).

6.3.3 Connection with Convolutional Layers

When a Plugin Network is attached to a convolutional layer, it adjusts the feature map obtained from a given convolutional filter. Thus, it has to compute vector $\mathbf{r}$, which has to be of the same dimension as the number of channels in the tensor $\mathbf{z}$. Then all the considered operators $\oplus$ in eq. (6.2) have elementwise meaning:

$$\tilde{\mathbf{z}}_c = \mathbf{z}_c \oplus r_c = \begin{bmatrix} z_{11} & \cdots & z_{1j} \\ \vdots & \ddots & \\ z_{i1} & & z_{ij} \end{bmatrix}_c \oplus r_c, \tag{6.4}$$

where c indicates the number of channels.

Adding a scalar value to each channel of a feature map greatly reduces the number of Plugin Network parameters that have to be learned. Learning different values for each elements of a feature map requires $w \times h \times c$ output values, where w, h stands for width and height of a feature map, respectively. Since we only add scalar value to each feature map, we require only c output values from a Plugin Network. Fig. 6.2 illustrates how Plugin Network is attached to convolutional layers.

6.3.4 Plugin Network Architecture

Overall, the Plugin Networks can be generalized to any model that can be trained with backpropagation. In our case, the Plugin Network is a FC neural network. Each FC layer is followed by a ReLU activation except for the last layer. We chose a fully connected architecture because partial evidence vector $\bar{\mathbf{y}}$ can be interpreted as a feature vector, which is used to compute a non-linear transform of outputs of the base model. This task can be well handled by a fully connected neural network.

We observe similar performance for each of the operators defined in eq. (6.2). Therefore, for consistency, we decide to always use the additive $\oplus$ operator throughout our experiments.

6.3.5 Training

The eq. (6.3) defines the joint model of a base network F with parameters $\mathbf{w}$ and the Plugin Network F_p with parameters $\mathbf{w}_p$. In the training procedure we are optimizing only $\mathbf{w}_p$

parameters, thus the base model F is not altered. To optimize the parameters of the Plugin Networks, the original loss function is used, *i.e.* the same loss function that is used to train the base model.

6.3.6 Properties

One important property of the Plugin Networks is that the function, which modifies a base model, is trained in an offline phase (see Section 6.3.5). Thus, the testing phase requires only a single feed-forward propagation through base model and the Plugin Networks. Thanks to the single forward pass, our model is fast, and forward propagation overhead of the Plugin Networks is negligible. Thus, our method is significantly faster than the model proposed in [128], where iterative optimization process is applied at the inference phase.

6.4 Results

In this section we evaluate the Plugin Networks on three challenging computer vision tasks. We consider a hierarchical, multi-label classification and semantic segmentation problems. To stay consistent with the previous work on these subjects, we conduct the experiments in the same setups as [52, 111, 128]. Therefore, we use SUN397 [133, 134], COCO'14 [79] and Pascal VOC 2011 [28] datasets, respectively.

6.4.1 Hierarchical Scene Categorization

We apply our method on SUN397 dataset [133, 134]. The dataset is annotated with three coarse categories, 16 general scene categories and 397 fine-grained scene categories. Our task is to classify fine-grained categories, given true values for coarse categories, as it was performed in Hu *et al.* [52] and Wang *et al.* [128]. Thus, coarse categories serve as the partial evidence. We follow same experimental setup as [52, 128]: we split dataset into train, validation, and test split with 50, 10 and 40 images per scene category. To allow fair comparison to [52, 128], we use the AlexNet [74] with Softmax trained on fine-grained categories. It will serve as the base model for the Plugin Networks in this experiment. To evaluate our method we compute mean average precision (mAP), multi-class accuracy (MC Acc) and intersection-over-union accuracy (IoU Acc).

Ablation study: Plugin Network can be attached to different layers of the base model. Furthermore, one can attach more than one Plugin Network simultaneously. In the study below we analyze different combinations of the above choices, the results are summarized in Table 6.1. The results show, that Plugin Network improve performance of the base model regardless to which layer it is attached. On the other hand, the performance gain differs between chosen layers. Thus, couple of observations can be drawn. It is more effective to attach a Plugin Network to an FC layer rather than a conv layer. Secondly, the Plugin Network connected to deeper layers tends to obtain better performance. The results are aligned with the intuition, that in case of the classification task, deeper layers carry more abstract information. Thus, the Plugin Network can converge to better solution when attached to deeper layers. Moreover, in classification task spatial information is ignored, thus modification of conv layers, which carry spatial information is less important. Note, that in case of semantic segmentation task described in Sec. 6.4.3, it is more important to modify conv layers. These experiments show that the Plugin Networks are generic and can solve various tasks.

TABLE 6.1: Performance of the Plugin networks w.r.t. the number of plugins and layers at which they are attached to the AlexNet. The comparison is done on SUN397 dataset. The performance is better if plugin is attached to deeper layers. Plugins attached to FC layers perform better. The most effective is plugin attached to fc3 layer, although it outperforms slightly models where three and six plugins were attached. We believe that it is characteristic for classification task.

Layer	MC Acc
no plugin	53.15
conv1	54.08
conv2	53.88
conv3	53.75
conv4	54.30
conv5	54.86
conv3-5	56.88

Layer	MC Acc
fc1	53.90
fc2	56.47
fc3	**57.51**
fc1-3	57.08
conv3-5, fc1-3	57.16

TABLE 6.2: Performance of the Plugin Network with respect to different number of hidden layers on SUN397. We consider two cases. Former is the Plugin Network being attached to the conv5 layer, while the latter — attached to the fc3 layer. The experiment shows that simple linear transformation (0 hidden layers) is not sufficient.

Layer	# of hidden layers				
	0	1	2	3	4
conv5	53.41	54.06	54.18	**54.28**	53.90
fc3	54.53	56.65	57.18	**57.51**	56.56

TABLE 6.3: Plugin Network performance on SUN397 dataset. Our method outperforms state-of-the-art on all reported metrics. To allow fair comparison, we also show the performance of base model used in [128] as well as the performance of base model trained by us.

	MC Acc	mAP	IoU Acc
Base model [128]	52.83±0.24	56.17±0.21	35.90±0.22
SINN [52, 128]	54.30±0.35	58.34±0.32	37.28±0.34
F. Prop (LF) [128]	54.94±0.42	58.52±0.34	37.86±0.39
F. Prop (RF) [128]	55.01±0.35	58.70±0.26	37.95±0.33
Base (Ours)	53.30±0.29	56.36±0.21	34.39±0.31
Plugin Net (Ours)	**57.59±0.24**	**61.55±0.43**	**39.26±0.38**

TABLE 6.4: Performance of the Plugin Network model trained on a fraction of training data. The results show that model trained even with 20% of available data, achieves state-of-the-art performance on all metrics. The results are reported on SUN397 dataset.

	% of training examples				
	20	40	60	80	100
MC Acc	56.58	57.07	57.26	57.34	57.51
mAP	60.75	61.19	61.26	61.27	61.37
IoU	38.98	39.39	39.42	39.60	39.62

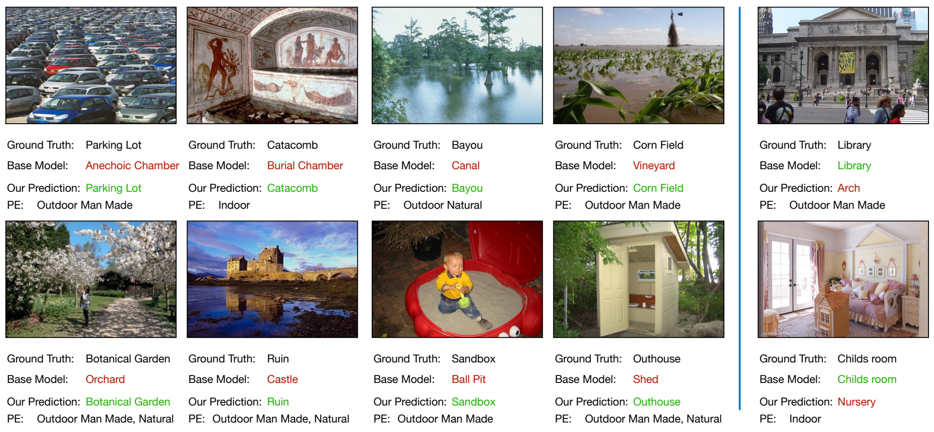
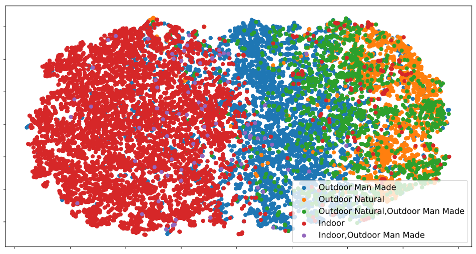


FIGURE 6.3: Visualization of results (best viewed in color). We pick 10 representative images from the SUN397 test set and visualize the predicted fine grained categories from our method. We compare them with the predictions from base model (CNN+Softmax). Correct predictions are marked in green, incorrect in red. Failure cases are shown in the rightmost column. "PE" stands for partial evidence.

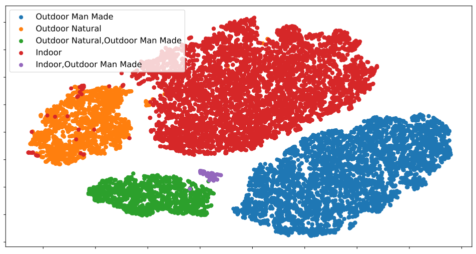
We do not observe further performance improvements, if more than one Plugin Network is attached simultaneously. In case of classification task, the Plugin Network mainly learn relationship between partial evidence and output labels. Thus, fc3 outputs carry enough information to find such a relationship.

In the second ablation study, we consider different Plugin Network architectures. We evaluate the number of hidden layers of the Plugin Network. We check from 0 to 4 hidden layers. The results are reported in Tab. 6.2. The best results are obtained by network with 3 hidden layers, which is still a quite shallow architecture that allows efficient training and do not add significant overhead in the inference. The results also show that a linear function (model with 0 hidden layers) has not enough capacity to adjust base model outputs.

In our model we use AlexNet CNN + Softmax as the base model. The CNN was pretrained on the Places365 dataset [137]. The base model is chosen to allow a fair comparison with [128, 52]. We use the Plugin Network with 3 hidden layers, attached to the fc3 layer from the base model. The model is trained for 15 epochs using the Adam [64] optimizer with learning rate set to $1e-3$. The learning rate is reduced to $1e-4$ and $1e-5$ after 5 and 10 epochs, respectively.



(a) Base Model



(b) Plugin Network

FIGURE 6.4: Comparison of representation learned by the base model and the Plugin Network. The output of the layer where Plugin Network is attached, is projected to 2 dimensions using t-SNE. Each point in the figure refers to one example from the test set of SUN397. Our model finds better separation w.r.t. partial evidence categories.

Training analysis: In Table 6.4 we report the performance of our method w.r.t. to the percentage of available training data. We train Plugin Network with 20%, 40%, 60%, and 80% percent of the training data. The results in Table 6.4 show that our model achieves the state-of-the-art performance even when trained on 20% of the data.

Comparison with the state-of-the-art: In Table 6.3 we report the performance of our Plugin Network. The results are averaged over 5 runs to mitigate the randomness in validation set sampling, also standard deviation is computed. We report performance of base model, which does not use partial evidence information as a reference. We also report performance of SINN network [52] and FeedbackProp [128]. The results in Table 6.3 show that Plugin Networks outperform state-of-the-art methods in terms of MC Acc, mAP and IoU. In addition, our method is easier to train than SINN model and allows faster inference than FeedbackProp.

Observations: In Figure 6.3 we show 10 images from the SUN397 test set. For each example we show: ground-truth label for fine-grained category, classification result from the base model together with result from our model. For each example we also report coarse category (partial evidence). The examples show that our method can recover many errors thanks to the presence of partial evidence information. For instance, in top left example, partial evidence that "parking lot" belongs to "Outdoor man made" category helped to correct the classification error. The base model classified example as "Anechoic Chamber", which belongs to "Indoor" category. If we look at the "Anechoic Chamber" chamber examples, one can notice that cars parked in the parking lot can mimic patterns on the Anechoic Chamber walls.

Representation analysis: we analyze how the intermediate output of base model was changed, after Plugin Network was added. First, we projected the activations of base model for each example in a test set to 2D plane using t-SNE method [82]. Figure 6.4 (A) Base Model shows the representation of the base model, while (B) Plugin Network shows the representation of our method. The figure shows that our model learns much better representation, as partial evidence categories are clearly separable. It is also interesting that our model manages to learn such representation, because the loss function that we optimize considers only fine grained labels and ignores error on partial evidence categories. The results show that our model learns dependency between partial evidence categories and fine-grained labels, while not being directly guided by the loss function.

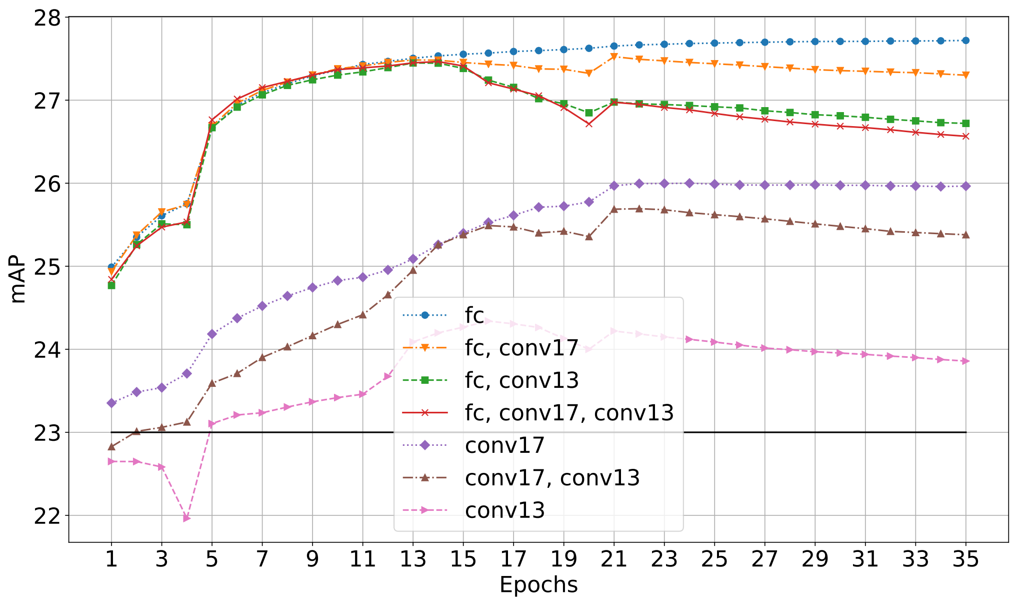


FIGURE 6.5: Performance of the base network ResNet-18 with the Plugin Networks attached to its different layers on the validation set. Plugins are attached to the last FC layer, 17th conv layer and 13th conv layer (13). The plugins have the same architecture, which is a two layered fully connected network with 500, and 2048 neurons. The black solid line at the bottom indicates mAP achieved by the base network with no plugins attached. Attachment to the last fc layer only results in the highest improvement.

We also tried to incorporate loss on partial evidence categories, but we did not observe an increase in model performance. Thus we decided to optimize loss function only on unknown labels. Such solution has another advantage – partial evidence categories do not have to be a subset of labels that base model recognizes.

6.4.2 Multi-label Image Annotation

In the evaluation of our method for the multi-label image annotation task, we use the COCO 2014 dataset [79]. It contains 120,000 images, each annotated with 5 caption sentences. Again, for consistency, we follow the same experimental setup as [128]. Namely, we use the provided 82,783 training data instances as our training set, and randomly split the remaining provided validation data into 20,000 validation set and 20,504 test set images.

TABLE 6.5: Results on the COCO'14. The baseline is ResNet-18 trained for the multi-label experiment. Our method not only achieves the state-of-the-art in means of the mAP, but also is the fastest during inference, being barely slower than the base network.

	mAP	Inference time [s]
Base model [30]	23.00	25.64
F. Prop (LF) [128]	25.26	93.36
F. Prop (RF) [128]	25.70	103.27
Plugins (Ours)	**27.97**	**25.72**

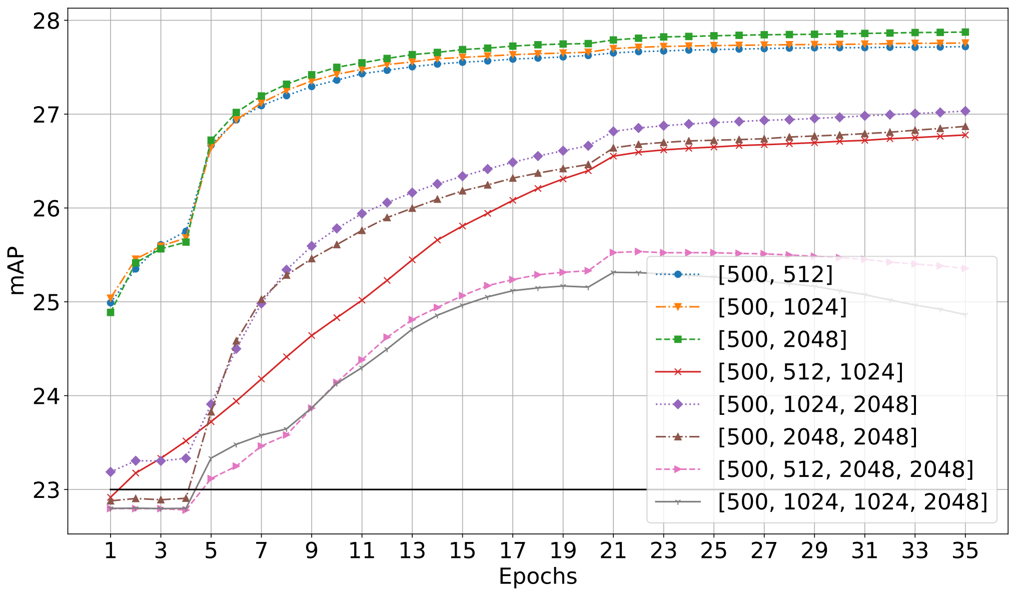


FIGURE 6.6: Performance of the base network ResNet-18 with plugin network attached to its last layer with different architectures of the plugin on the validation set. Numbers in brackets indicate the number of neurons at consecutive layers. The black solid line at the bottom indicates mAP achieved by ResNet-18 no plugins attached. If the plugin has too many layers, it starts to overfit. We report the highest performance by using a two-layered network.

TABLE 6.6: Scores obtained when attaching plugins at different places to different versions of ResNet. We consider attachments to the last FC layer, end of the third residual layer (RL3) and fourth residual layer (RL4). Plugins always consist of 500 and 2048 neurons. For this task, we always achieve the highest score when applying a single plugin to the last layer of the base network.

	no plugin	FC, RL4, RL3	FC, RL4	**FC**
ResNet18	23.00	27.56	27.61	**27.97**
ResNet50	25.84	29.85	29.65	**29.93**
ResNet101	26.56	29.49	29.79	**30.13**

The task is to predict a predefined set of words explaining an image. These words are referred as visual concepts in the work of Fang *et al*. in their visual concept classifier [30]. We define them as the 1,000 most frequent words in the captions of the COCO dataset. We use the same tokenization, lemmatization, and stop-word removals as Wang *et al*. [128]. As a result, each image is annotated by a vector of 1,000 elements corresponding to an occurrence of words in the captions.

For the task of reasoning under partial evidence, we randomly divide the target vector into a fixed 500 known and 500 unknown classes. The comparison is performed on the unknown set only, while the known set is used as the partial evidence. The base network is first trained as in Fang *et al*. [30] on the multi-labeled task using the entire 1,000 classes. It is done by minimization of the binary cross entropy between the predicted and target vector of concepts. For this experiment, for the base network we choose to use the ResNet-18 architecture [47] to stay consistent with [128]. Additionally, we also make an ablation study for deeper base network architectures: ResNet-50 and ResNet-101.

Hyperparameters selection: The architecture of the plugin is chosen based on the validation scores from Figs. 6.5 and 6.6. As indicated before, we first train the base network on the given task. Then, we freeze the base network's weights and add a number of plugins. We train each plugin for 36 epochs with a starting learning rate of 10^{-3}, which decrease with the number of epochs to 10^{-4}. We use the Adam [64] optimizer with the Xavier initialization [41].

During the hyperparameters selection, we verify all combinations of attaching a plugin to the conv13, conv17 and FC layers of the ResNet-18 network. We report, that a single attachment to the last FC layers results in the highest mAP. Using any earlier layer still improves the baseline, but such plugin overfits much easier. Using a combination of the FC layer and any other convolutional layer leads to decrease of the performance comparing to a single attachment to the FC layer due to the overfitting; see Fig. 6.5. This results are aligned with the conclusions drawn in ablation study from Section 6.4.1.

In next experiment we search for the best architecture of a single plugin. We consider different number of layers and neurons in the Plugin Networks. See Fig. 6.6 for details. The highest score is achieved by using the two layered architectures with 500 and 2,048 neurons in a layer, respectively. The two and three layered networks get to the plateau after around 20 epochs, while the four-layered networks start to overfit.

Comparison with the state-of-the-art: We compare ourselves to the Layer-wise Feedback-prop (LF) and Residual Feedback-prop (RF) Inference proposed by [128]. Results presented in this work are based on the open sourced online implementation[1] provided by the authors of RF and LF. Due to the random choice of the known and unknown labels, the baseline may differ, but the overall gain stays similar. We show, that in terms of the mAP, we achieve the state-of-the-art with a significant margin. Furthermore, as expected, the inference phase is much faster compared to the Feedback-prop methods. Please refer to Tab. 6.5 for results.

Finally, we verify the usage of the Plugin Networks for deeper architectures, such as ResNet50 and ResNet101; see Tab. 6.6. We notice an improvement for each base network. As for the ResNet-18, we achieve the highest score when only one plugin is used at the last FC layer of the base network.

[1]`github.com/uvavision/feedbackprop`

TABLE 6.7: Scores obtained when attaching plugins at different places to the FCN architecture for the task of semantic segmentation. conv stands for a convolutional and deconv for transposed convolutional layers.

Model	number of plugins	mean-IoU
Baseline	0	65.5
conv1-3	3	65.7
conv1-5	5	70.5
deconv1-3	3	71.1
deconv1-5	5	71.2
conv1-5, deconv1-5	10	**72.2**

6.4.3 Scene semantic segmentation

In this section we evaluate the Plugin Networks on multi-cue object class segmentation task. We take fully convolutional network (FCN) [118] as a starting point. Next, we experiment by adding several plugins into its architecture. For the baseline we take the pre-trained FCN-8s model[2] trained on the SBD dataset [45]. We use the same dataset when training the Plugin Networks. We validate our models on the Pascal VOC 2011 segmentation challenge dataset [28]. We follow [118] and take *Pascal VOC 2011 segval*[3] validation split in order to avoid overlapping images between these two datasets. Thus, our training and validation datasets consist of 8,498 and 736 images, respectively. Objects in the image are assigned to one of 21 classes.

The base model (FCN-8s without plugins) results in IoU score of 65.5%. For this scenario, we assume that we have the knowledge of classes present in the image at the inference time, which constitutes partial evidence. Therefore, our partial evidence is a vector of 21 elements. The goal of the Plugin Network is to improve the output segmentation masks of the base network. We experiment with attaching several plugins to the FCN-8s model. We report the results in Tab. 6.7. In contrary to findings from previous experiments, the Plugin Networks provide the highest increase in the IoU, when multiple of them are used. We achieve the highest gain of 72.2% of IoU when attaching five plugins into all of the convolutional layers and all of the transposed convolutional layers. We also outperform the previously proposed method [111], which achieved 69.2%, on the partial evidence task for semantic segmentation.

Using partial evidence through the Plugin Networks, we are able to soften the wrong feature maps and strengthen the expected ones. It results in major improvement of the base network. When multiple objects are present in the scene, the base model may have a problem to consistently assign a proper label to a particular object. As expected, when the baseline network makes a mistake by assigning a wrong label of a class that is not present in the image, plugins correct these with a correct class. The examples are shown in Figure 6.7. For instance in

[2] `github.com/wkentaro/pytorch-fcn`

[3] `github.com/shelhamer/fcn.berkeleyvision.org`

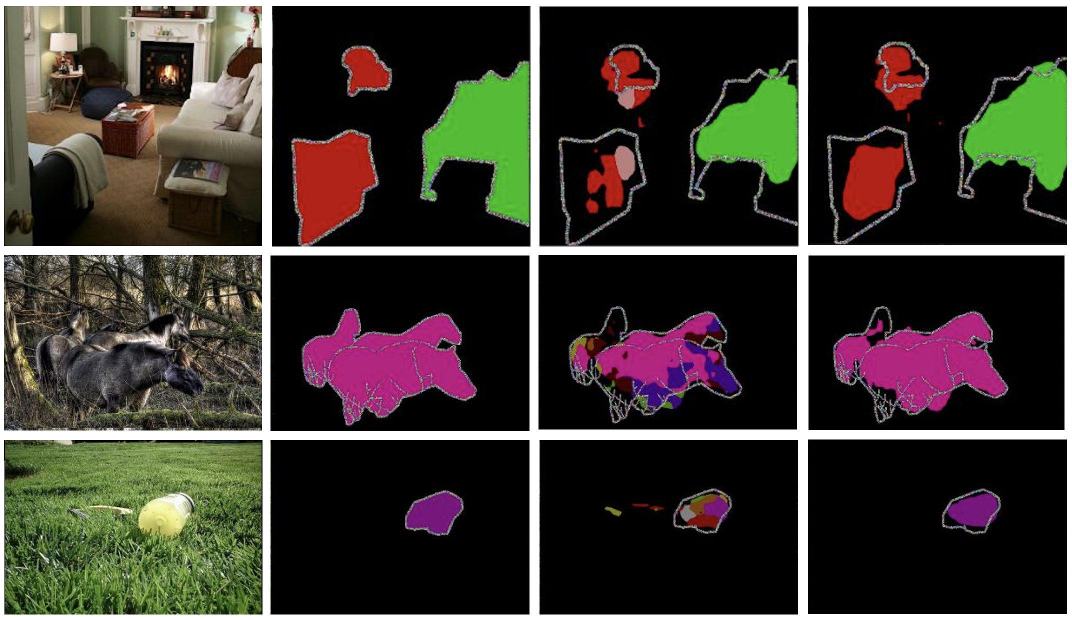


FIGURE 6.7: Examples of semantic segmentation masks predicted by the base model with and without Plugin Networks. The images are taken from the Pascal VOC 2011 validation set. In rows we show different examples, and in columns, from left to right: input image, ground-truth (all classes), base model, FCN-8s with 10 plugins. Our Plugin Networks show clear improvement of the base model output segmentation masks.

the last row, the pixels belonging to a bottle are inconsistently assigned to different classes by the base model. Using the Plugin Networks fixes the problem.

6.5 Conclusion

In this work, we introduce the Plugin Networks – a simple, yet effective method to exploit the availability of a partial evidence in the context of visual recognition tasks. Plugin Networks are integrated directly with the intermediate layers of pre-trained convolutional neural networks and thanks to their lightweight design can be trained efficiently with low computational cost and limited amount of data. Results presented on three challenging tasks and various datasets show superior performance of the proposed method with respect to the state-of-the-art approaches.

We experiment with multiple configurations of our system. We focus especially on: number of plugins, plugin depth, fusion layer type (conv or fc), amount of training data and fusion location. Regardless of the fusion operator choice, our method always outperforms the state-of-the-art methods. While our modulations of the base-network, seems to be similar to the Priming Neural Networks or FiLM, they are significantly different. The main difference, is the idea of using an additional deep neural network to learn the modulation function in contrary to a simple linear model. It might occur trivial, but it is a necessary step towards improving state-of-the-art by a large margin in our scenario. We show in our ablation studies, that a single-layered network would not be able to improve the results sufficiently. Plugin Network alters only base model activations, while methods like FiLM requires incorporation of new convolutional layers into the base network architecture. Plugin Network changes neither the base model architecture nor changes the weights and alters only the activations. In the case of Feedback-prop the partial evidence labels can only be a subset of labels that the base model can recognize. Our method, however, can accept any kind of labels as partial evidence. Moreover, the Feedback-prop by design requires a multiple forward and backward passes during the inference in contrast to a single forward pass for the Plugin Networks. This results in a much shorter inference time required by the Plugin Networks.

If baseline model prediction scores are close for two classes, *e.g. library* and *arch*, and the partial evidence is the same for both, *e.g. outdoor, man made*, Plugin Network can add some unwanted noise, which causes a failure. This can be seen in the examples from 6.3. We reason that one issue causing the problem is the fact that the Plugin Networks do not take advantage of the input source and the resulting output and in the consequence are blind to them. It is very likely that the contextual information would boost the decision process of a Plugin Network. Another possibility would be to add an iterative, or recurrent channel to allow the Plugin Network to reason from the resulting, merged output of the base model. We leave these aspects for the future investigation and research.

Our Plugin Networks are agnostic to the input signal and can accommodate arbitrary modality of the input data, including audio or textual cues. Therefore, their multi modal nature can allow richer contextual cues to be taken into account in the inference procedure, leading to more effective and efficient visual recognition models. Currently, neural models while trained are very difficult to modify. Adding new output (*e.g.* new classes) or new input (*e.g.* partial evidence) requires training from scratch or via fine-tuning. If a base model is fine-tuned, phenomena such as the catastrophic forgetting arise. These limitations are overlooked by many authors, who focus on uni-modal input (*e.g.* images). In many scenarios data from other sources is available (GPS, text). Our method allows an easy adaption of a base model to available partial evidence. We believe that this work can open novel research directions related to solving visual recognition tasks with partial evidence.

Chapter 7

Conclusion

In this thesis we develop and discuss machine and deep learning techniques for semantic and instance segmentation. The techniques were evaluated on a dataset of CT scans of short glass fiber reinforced polymers prepared in cooperation with the University of Padova and on publicly available medical CT scans of lungs and liver. The last chapter on Plugin Networks evaluates on a public and popular large-scale object detection, segmentation, and captioning dataset for a better comparison with the state-of-the-art.

In chapters 2, 3 and, 4 we present our work on fiber segmentation from short glass fiber reinforced polymer X-ray CT scans. Chapter 2 describes the data acquisition and dataset preparation of the reference volumes from the part manufacturing to computational model of SFRP. In Chapter 3 and 4 we propose state-of-the-art models for semantic and instance segmentation of fibers from SFRP. We evaluate them on our challenging dataset of low resolution CT scans. In chapter 5 we present applications of machine and deep learning models for real-life applications in the medical imaging field. We propose a 3D dictionary learning model for bronchial vessel segmentation achieving the state-of-the-art on the VESsel12 dataset. For the liver and liver lesion segmentation from CT scans we propose a U-net based architecture and the *Mask Mining* boosting technique for a wide array of machine learning techniques. In chapter 6 we present the *Plugin Networks* – a solution for inference under partial evidence. We evaluate it on the general tasks of hierarchical scene categorization, multi-label image annotation and scene semantic segmentation achieving state-of-the-art on each.

Future work on segmentation should focus on instance segmentation, most importantly on different embedding regularization together with unsupervised clustering methods. For instance it would be interesting to see the use of Distance Matching Regularizers or Adjusted Binarization Representation Entropy Regularizer on the embedding space as has been used for generative adversarial networks [138]. Use of described methods on different applications would be of great interest too to see how the algorithms behave for more complicated structures. As for Plugin Networks the future work should focus on allowing the plugins to gather context of the input source in some form. For instance via embedding learning or iterative, recurrent approach.

Appendix A

List of Peer Reviewed Publications

M Koperski*, **T Konopczyński***, R Nowak, P Semberecki, T Trzciński. Plugin Networks for Inference under Partial Evidence. Winter Conference on Applications of Computer Vision (WACV). 2020 *Equal contribution

JS Rathore, **T Konopczyński**, J Hesser, G Lucchetta, Simone Carmignato. Investigation on Tomographic Based NDT Characterization of Short Glass Fiber Reinforced Composites as Obtained from Micro Injection Molding. Journal of Nondestructive Evaluation, Diagnostics and Prognostics of Engineering Systems (JNDE). 2020

K Roth, J Hesser, **T Konopczyński**. Mask Mining for Improved Liver Lesion Segmentation. International Symposium on Biomedical Imaging (ISBI). 2020

E Fotiadou, **T Konopczyński**, J Hesser, R Vullings. End-to-End Trained CNN Encoder-Decoder Network for Fetal ECG Signal Denoising. Physiological Measurement. 2020

E Fotiadou, **T Konopczyński**, J Hesser, R Vullings. Deep Convolutional Encoder-Decoder Framework for Fetal ECG Signal Denoising. Computing In Cardiology (CiC). 2019

T Konopczyński, T Kröger, L Zheng, J Hesser. Instance Segmentation of Fibers from Low Resolution CT Scans via 3D Deep Embedding Learning. The British Machine Vision Conference (BMVC). 2018

T Konopczyński, D Rathore, J Rathore, T Kröger, L Zheng, CS Garbe, J Hesser. Fully Convolutional Deep Network Architectures for Automatic Short Glass Fiber Semantic Segmentation from CT Scans. The e-Journal of Nondestructive Testing. 2018

T Konopczyński, J Rathore, T Kröger, L Zheng, CS Garbe, S Carmignato, J Hesser. Reference Setup for Quantitative Comparison of Segmentation Techniques for Short Glass Fiber CT Data. The e-Journal of Nondestructive Testing. 2017

T Konopczyński, T Kröger, L Zheng, CS Garbe, J Hesser. Automated multiscale 3D feature learning for vessels segmentation in Thorax CT images. IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD). 2016

D Prodanov, **T Konopczyński**, M Trojnar. Selected Applications of Scale Spaces in Microscopic Image Analysis. Cybernetics and Information Technologies. 2015

Bibliography

[1] Esteva A et al. “Dermatologist-level classification of skin cancer with deep neural networks.” In: *Nature* (2017).

[2] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `http://tensorflow.org/`.

[3] Daniele Annicchiarico, Usama M Attia, and Jeffrey R Alcock. “A methodology for shrinkage measurement in micro-injection moulding”. In: *Polymer testing* 32.4 (2013), pp. 769–777.

[4] M. Bai and R. Urtasun. “Deep watershed transform for instance segmentation”. In: *Conference on Computer Vision and Pattern Recognition* (2017), pp. 2858–2866.

[5] Pierre Baldi. “Autoencoders, unsupervised learning, and deep architectures”. In: *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012, pp. 37–49.

[6] Carsten Bellon et al. “Radiographic simulator aRTist: version 2”. In: *18th World Conference on Nondestructive Testing, Durban, South Africa, 16–20 April 2012*. Citeseer. 2012.

[7] Andrea Bernasconi, Francesca Cosmi, and PJ Hine. “Analysis of fibre orientation distribution in short fibre reinforced polymers: A comparison between optical and tomographic methods”. In: *Composites Science and Technology* 72.16 (2012), pp. 2002–2008.

[8] Patrick Bilic et al. “The Liver Tumor Segmentation Benchmark (LiTS)”. In: *CoRR* abs/1901.04056 (2019).

[9] Kristina Bliznakova et al. “Modelling of small CFRP aerostructure parts for X-ray imaging simulation”. In: *International Journal of Structural Integrity* 5.3 (2014), pp. 227–240.

[10] B. De Brabandere, D. Neven, and L. Van Gool. “Semantic instance segmentation with a discriminative loss function”. In: *arXiv preprint arXiv:1708.02551.* (2017).

[11] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[12] Jack E Bresenham. “Algorithm for computer control of a digital plotter”. In: *IBM Systems journal* 4.1 (1965), pp. 25–30.

[13] *BS EN ISO 294-3: 2003, Plastics – Injection Moulding of Test Specimens of Thermoplastic Materials - Part 3: Small Plates.*

[14] S Carmignato, A Voltan, and E Savio. “Metrological performance of optical coordinate measuring machines under industrial conditions”. In: *CIRP annals* 59.1 (2010), pp. 497–500.

[15] S Carmignato et al. “Testing of x-ray microtomography systems using a traceable geometrical standard”. In: *Measurement Science and Technology* 20.8 (2009), p. 084021.

[16] Simone Carmignato. “Accuracy of industrial computed tomography measurements: Experimental results from an international comparison”. In: *CIRP annals* 61.1 (2012), pp. 491–494.

[17] Simone Carmignato, Wim Dewulf, and Richard Leach. *Industrial X-ray computed tomography*. Springer, 2018.

[18] Grzegorz Chlebus et al. "Neural Network-Based Automatic Liver Tumor Segmentation With Random Forest-Based Candidate Filtering". In: (2017). URL: `arXiv:1706.00842`.

[19] P. F. Christ et al. "Automatic liver and lesion segmentation in CT using cascaded fully convolutional neural networks and 3D conditional random fields". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2016), pp. 415–423.

[20] Özgün Çiçek et al. "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation". In: *CoRR* abs/1606.06650 (2016).

[21] Adam Coates and Andrew Y Ng. "The importance of encoding versus training with sparse coding and vector quantization". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 921–928.

[22] *Consortium for Open Medical Image Computing. http://grand-challenge.org/. Accessed: 2016-12-15.*

[23] Jia 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.

[24] Jia Deng et al. "Large-Scale Object Classification Using Label Relation Graphs". In: *European Conference on Computer Vision*. 2014.

[25] Santosh Kumar Divvala et al. "An empirical study of context in object detection". In: *CVPR*. IEEE Computer Society, 2009, pp. 1271–1278.

[26] Michal Drozdzal et al. "The Importance of Skip Connections in Biomedical Image Segmentation". In: *CoRR* abs/1608.04117 (2016).

[27] M. Ester et al. "Density-based spatial clustering of applications with noise". In: *Int. Conf. Knowledge Discovery and Data Mining* (1996).

[28] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results*. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[29] Chollet F. "Keras: Deep Learning library for Theano and TensorFlow". In: (2017).

[30] Hao Fang et al. "From captions to visual concepts and back." In: *CVPR*. IEEE Computer Society, 2015, pp. 1473–1482. ISBN: 978-1-4673-6964-0. URL: `http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#FangGISDDGHMPZZ15`.

[31] T Fast et al. "Topological and Euclidean metrics reveal spatially nonuniform structure in the entanglement of stochastic fiber bundles". In: *Journal of materials science* 50.6 (2015), pp. 2370–2398.

[32] A. Fathi et al. "Semantic instance segmentation via deep metric learning". In: *arXiv preprint arXiv:1703.10277.* (2017).

[33] Amir H Foruzan et al. "A Hessian-based filter for vascular segmentation of noisy hepatic CT scans". In: *International journal of computer assisted radiology and surgery* 7.2 (2012), pp. 199–205.

[34] Eleni Fotiadou et al. "Deep Convolutional Encoder-Decoder Framework for Fetal ECG Signal Denoising". In: *Computing in Cardiology* (2019).

[35] Eleni Fotiadou et al. "End-to-End Trained CNN Encoder-Decoder Network for Fetal ECG Signal Denoising". In: *Physiological Measurement* (2020).

[36] Alejandro F Frangi et al. "Multiscale vessel enhancement filtering". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 1998, pp. 130–137.

[37] Shao-Yun Fu and Bernd Lauke. “Effects of fiber length and fiber orientation distributions on the tensile strength of short-fiber-reinforced polymers”. In: *Composites Science and Technology* 56.10 (1996), pp. 1179–1190.

[38] Shao-Yun Fu, Bernd Lauke, and Y-W Mai. *Science and engineering of short fibre reinforced polymer composites*. Elsevier, 2009.

[39] Shao-Yun Fu and Yiu-Wing Mai. “Thermal conductivity of misaligned short-fiber-reinforced polymer composites”. In: *Journal of applied polymer science* 88.6 (2003), pp. 1497–1505.

[40] Carolina Galleguillos, Andrew Rabinovich, and Serge J. Belongie. “Object categorization using co-occurrence, location and appearance”. In: *CVPR*. IEEE Computer Society, 2008.

[41] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[42] Kristen Grauman, Fei Sha, and Sung Ju Hwang. “Learning a Tree of Metrics with Disjoint Visual Features”. In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 621–629. URL: `http://papers.nips.cc/paper/4250-learning-a-tree-of-metrics-with-disjoint-visual-features.pdf`.

[43] K Hameeteman et al. “Evaluation framework for carotid bifurcation lumen segmentation and stenosis grading”. In: *Medical image analysis* 15.4 (2011), pp. 477–488.

[44] Xiao Han. “Automatic Liver Lesion Segmentation Using A Deep Convolutional Neural Network Method”. In: *CoRR* abs/1704.07239 (2017). URL: `http://arxiv.org/abs/1704.07239`.

[45] B. Hariharan et al. “Semantic contours from inverse detectors”. In: *2011 International Conference on Computer Vision*. 2011, pp. 991–998. DOI: `10.1109/ICCV.2011.6126343`.

[46] K. He et al. “Mask r-cnn”. In: *International Conference on Computer Vision* (2017), pp. 2980–2988.

[47] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[48] Kaiming He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *CoRR* abs/1502.01852 (2015). URL: `http://arxiv.org/abs/1502.01852`.

[49] Moritz Helmstaedter, Kevin L Briggman, and Winfried Denk. “High-accuracy neurite reconstruction for high-throughput neuroanatomy”. In: *Nature neuroscience* 14.8 (2011), p. 1081.

[50] Petr Hermanek et al. “Principles of X-ray computed tomography”. In: *Industrial X-Ray Computed Tomography*. Springer, 2018, pp. 25–67.

[51] Jiang Hsieh et al. “Computed tomography: principles, design, artifacts, and recent advances”. In: SPIE Bellingham, WA. 2009.

[52] Hexiang Hu et al. “Learning structured inference neural networks with label relations”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2960–2968.

[53] Jie Hu et al. “Squeeze-and-Excitation Networks”. In: *CoRR* abs/1709.01507 (2017).

[54] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016). URL: `http://arxiv.org/abs/1608.06993`.

[55] Lawrence Hubert and Phipps Arabie. "Comparing partitions". In: *Journal of classification* 2.1 (1985), pp. 193–218.

[56] Sung Ju Hwang, Kristen Grauman, and Fei Sha. "Semantic Kernel Forests from Multiple Taxonomies". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1718–1726. URL: `http://papers.nips.cc/paper/4760-semantic-kernel-forests-from-multiple-taxonomies.pdf`.

[57] Arganda-Carreras I et al. "Crowdsourcing the creation of image segmentation algorithms for connectomics". In: *Frontiers in neuroanatomy* (2015).

[58] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015).

[59] Fabian Isensee et al. "nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation". In: *CoRR* abs/1809.10486 (2018).

[60] Gerd-Rüdiger Jaenisch, Carsten Bellon, and Uwe Ewert. "aRTist–Analytical RT inspection simulation tool for industrial application". In: *Proceedings of the 17th World Conference on Non-Destructive Testing, Shanghai, China, International Committee on NDT, CDrom paper*. Vol. 64. Citeseer. 2008.

[61] Justin Johnson, Lamberto Ballan, and Li Fei-Fei. "Love Thy Neighbors: Image Annotation by Exploiting Image Metadata". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 4624–4632.

[62] Simon Jégou et al. "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation". In: *CoRR* abs/1611.09326 (2016). URL: `http://arxiv.org/abs/1611.09326`.

[63] J. Kastner, B. Plank, and D. Salaberger. "High resolution X-ray computed tomography for quantitative characterization of fiber reinforced polymers and heterogeneous light metals". In: *Proceeding ASNT 22nd Annual Research Symposium*. 2013.

[64] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[65] Michael Kinsella et al. "Mechanical properties of polymeric composites reinforced with high strength glass fibers". In: *International SAMPE Technical Conference*. Vol. 33. 2001, pp. 1644–1657.

[66] Ryan Kiros et al. "Stacked multiscale feature learning for domain independent medical image segmentation". In: *International workshop on machine learning in medical imaging*. Springer. 2014, pp. 25–32.

[67] U Koethe. *Computer vision library vigra*. 2010.

[68] Tomasz Konopczyński et al. "Automated multiscale 3D feature learning for vessels segmentation in Thorax CT images". In: *IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)*. IEEE. 2016, pp. 1–3. URL: `https://arxiv.org/abs/1901.01562`.

[69] Tomasz Konopczyński et al. "Reference Setup for Quantitative Comparison of Segmentation Techniques for Short Glass Fiber CT Data". In: *Conference on Industrial Computed Tomography* (2017). URL: `https://arxiv.org/abs/1901.01210`.

[70] Tomasz Konopczyński et al. "Fully Convolutional Deep Network Architectures for Automatic Short Glass Fiber Semantic Segmentation from CT scans". In: *Conference on Industrial Computed Tomography* (2018). URL: https://arxiv.org/abs/1901.01211.

[71] Tomasz Konopczyński et al. "Instance Segmentation of Fibers from Low Resolution CT Scans via 3D Deep Embedding Learning". In: *British Machine Vision Conference (BMVC)* (2018). URL: https://arxiv.org/abs/1901.01034.

[72] Michal Koperski et al. "Plugin Networks for Inference under Partial Evidence". In: *Winter Conference on Applications of Computer Vision (WACV)* (2020). URL: https://arxiv.org/abs/1901.00326.

[73] Karl Krissian et al. "Model-based detection of tubular structures in 3D images". In: *Computer vision and image understanding* 80.2 (2000), pp. 130–171.

[74] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[75] Jean Pierre Kruth et al. "Computed tomography for dimensional metrology". In: *CIRP annals* 60.2 (2011), pp. 821–842.

[76] François Lanusse et al. "CMU DeepLens: deep learning for automatic image-based galaxy–galaxy strong lens finding". In: *Monthly Notices of the Royal Astronomical Society* 473.3 (2017), pp. 3895–3906.

[77] Sang-Woo Lee et al. "Overcoming Catastrophic Forgetting by Incremental Moment Matching". In: *CoRR* abs/1703.08475 (2017).

[78] Xi-Ping Li, Guo-Qun Zhao, and Can Yang. "Effect of mold temperature on motion behavior of short glass fibers in injection molding process". In: *The International Journal of Advanced Manufacturing Technology* 73.5-8 (2014), pp. 639–645.

[79] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1.

[80] G Lucchetta et al. "Investigating the technological limits of micro-injection molding in replicating high aspect ratio micro-structured surfaces". In: *CIRP Annals* 63.1 (2014), pp. 521–524.

[81] L. V. D. Maaten and G. Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9 (2008), pp. 2579–2605.

[82] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[83] Julien Mairal et al. "Supervised dictionary learning". In: *Advances in neural information processing systems*. 2009, pp. 1033–1040.

[84] Francesco Marinello et al. "Calibration artefact for the microscale with high aspect ratio: The fiber gauge". In: *CIRP annals* 57.1 (2008), pp. 497–500.

[85] Davide Masato, Marco Sorgato, and Giovanni Lucchetta. "Analysis of the influence of part thickness on the replication of micro-structured surfaces by injection molding". In: *Materials & Design* 95 (2016), pp. 219–224.

[86] Davide Masato et al. "Analysis of the shrinkage of injection-molded fiber-reinforced thin-wall parts". In: *Materials & Design* 132 (2017), pp. 496–504.

[87] Julian McAuley and Jure Leskovec. "Image Labeling on a Network: Using Social-Network Metadata for Image Classification". In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 828–841. ISBN: 978-3-642-33765-9.

[88] G Meneghetti et al. "An hysteresis energy-based synthesis of fully reversed axial fatigue behaviour of different polypropylene composites". In: *Composites Part B: Engineering* 65 (2014), pp. 17–25.

[89] Antoine Miech, Ivan Laptev, and Josef Sivic. "Learnable pooling with Context Gating for video classification". In: *CoRR* abs/1706.06905 (2017).

[90] Roozbeh Mottaghi et al. "The Role of Context for Object Detection and Semantic Segmentation in the Wild". In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 891–898. ISBN: 978-1-4799-5118-5. DOI: `10.1109/CVPR.2014.119`. URL: `https://doi.org/10.1109/CVPR.2014.119`.

[91] D. Neven et al. "Fast Scene Understanding for Autonomous Driving". In: *arXiv preprint arXiv:1708.02550* (2017).

[92] *Nikon MCT22 for Metrology CT Absolute accuracy for inside geometry (brochure). Accessed: 2016-12-15.* URL: `http://www.nikonmetrology.com/en\_EU/Products/X-ray-and-CT-Inspection/Metrology-CT/MCT225-for-Metrology-CT-Absolute-accuracy-for-inside-geometry/`.

[93] Hwa Jin Oh et al. "Warpage analysis of a micro-molded parts prepared with liquid crystalline polymer based composites". In: *Composites Part A: Applied Science and Manufacturing* 53 (2013), pp. 34–45.

[94] Sinue Ontiveros et al. "Dimensional measurement of micro-moulded parts by computed tomography". In: *Measurement Science and Technology* 23.12 (2012), p. 125401.

[95] Vicente Ordonez et al. "From Large Scale Image Categorization to Entry-Level Categories". In: *Proceedings of the 2013 IEEE International Conference on Computer Vision*. ICCV '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 2768–2775. ISBN: 978-1-4799-2840-8. DOI: `10.1109/ICCV.2013.344`. URL: `http://dx.doi.org/10.1109/ICCV.2013.344`.

[96] Nobuyuki Otsu. "A threshold selection method from gray-level histograms". In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[97] Ahmed N Oumer and Othman Mamat. "A review of effects of molding methods, mold thickness and other processing parameters on fiber orientation in polymer composites". In: *Asian Journal of Scientific Research* 6.3 (2013), p. 401.

[98] John F O'Gara, Glen E Novak, and MG Wyzgoski. "Predicting the tensile strength of short glass fiber reinforced injection molded plastics". In: *Proceedings of the 10th-Annual SPE® Automotive Composites Conference & Exhibition (ACCE), Troy, MI, USA*. 2010, pp. 15–16.

[99] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[100] Christ PF et al. "Automatic Liver and Tumor Segmentation of CT and MRI Volumes using Cascaded Fully Convolutional Neural Networks". In: (2017).

[101] Pascal Pinter, Benjamin Bertram, and Kay André Weidenmann. "A novel method for the determination of fibre length distributions from μct-data". In: *Conference on Industrial Computed Tomography (iCT)*. 2016.

[102] Zsuzsanna Püspöki et al. "Transforms and operators for directional bioimage analysis: a survey". In: *Focus on Bio-Image Informatics*. Springer, 2016, pp. 69–93.

[103] Jitendra Singh Rathore et al. "Effect of process parameters on dimensional accuracy of fiber-reinforced thin-walled micro moulded part". In: *17th International Conference of the European Society for Precision Engineering and Nanotechnology, Hannover, Germany* (May 2017), pp. 347–348.

[104] Jitendra Singh Rathore et al. "Investigation on tomographic based NDT characterization of short glass fiber reinforced composites as obtained from micro injection molding". In: *Journal of Nondestructive Evaluation, Diagnostics and Prognostics of Engineering Systems (JNDE)* (2020).

[105] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. "Learning multiple visual domains with residual adapters". In: *NIPS*. 2017, pp. 506–516.

[106] S. Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* (2015), pp. 91–99.

[107] Mina Rezaei et al. "Conditional Generative Refinement Adversarial Networks for Unbalanced Medical Image Semantic Segmentation". In: *CoRR* abs/1810.03871 (2018).

[108] Thomas Riedel. "Evaluation of 3D fiber orientation analysis based on x-ray computed tomography data". In: *Conference on Industrial Computed Tomography (ICT)*. 2012.

[109] B. Romera-Paredes and P. H. S. Torr. "Recurrent instance segmentation". In: *European Conference on Computer Vision* (2016), pp. 312–329.

[110] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[111] Amir Rosenfeld, Mahdi Biparva, and John K. Tsotsos. "Priming Neural Networks". In: *CVPR Workshops*. 2018.

[112] Karsten Roth, Jürgen Hesser, and Tomasz Konopczyński. "Boosting Liver and Lesion Segmentation from CT Scans By Mask Mining". In: *International Symposium on Biomedical Imaging* (2020). URL: `https://arxiv.org/abs/1908.05062`.

[113] Karsten Roth, Tomasz K. Konopczynski, and Jürgen Hesser. "Liver Lesion Segmentation with slice-wise 2D Tiramisu and Tversky loss function". In: *CoRR* abs/1905.03639 (2019). arXiv: `1905.03639`. URL: `http://arxiv.org/abs/1905.03639`.

[114] Rina D Rudyanto et al. "Comparing algorithms for automated vessel segmentation in computed tomography scans of the lung: the VESSEL12 study". In: *Medical image analysis* 18.7 (2014), pp. 1217–1232.

[115] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. "Tversky loss function for image segmentation using 3D fully convolutional deep networks". In: *CoRR* abs/1706.05721 (2017). URL: `http://arxiv.org/abs/1706.05721`.

[116] R Selden. "Thin wall modling of engineering plastics–a literature survey". In: *Journal of Injection Molding Technology* 4.4 (2000), p. 159.

[117] R. M. Sencu et al. "Generation of micro-scale finite element models from synchrotron X-ray CT images for multidirectional carbon fibre reinforced composites". In: *Composites Part A: Applied Science and Manufacturing* 91 (2016), pp. 85–95.

[118] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.4 (2017), pp. 640–651. DOI: `10.1109/TPAMI.2016.2572683`. URL: `https://doi.org/10.1109/TPAMI.2016.2572683`.

[119] Hongbin Shen, Steven Nutt, and David Hull. "Direct observation and measurement of fiber architecture in short fiber-polymer composite foam through micro-CT imaging". In: *Composites science and technology* 64.13-14 (2004), pp. 2113–2120.

[120] Marco Sorgato, Davide Masato, and Giovanni Lucchetta. "Effect of vacuum venting and mold wettability on the replication of micro-structured surfaces". In: *Microsystem Technologies* 23.7 (2017), pp. 2543–2552.

[121] Thanh Binh Nguyen Thi et al. "Measurement of fiber orientation distribution in injection-molded short-glass-fiber composites using X-ray computed tomography". In: *Journal of Materials Processing Technology* 219 (2015), pp. 1–9.

[122] Konstantinos Tigkos et al. "Simulation study for optimization of X-ray inspection setup applied to CFRP aerostructures". In: *ICT Conference*. Vol. 2014. 2014.

[123] David A Van Valen et al. "Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments". In: *PLoS computational biology* 12.11 (2016), e1005177.

[124] *VGStudio Max 3.0, http://www.volumegraphics.com/en/products/vgstudio-max/basic-functionality. Accessed: 2016-12- 15.*

[125] Luc Vincent and Pierre Soille. "Watersheds in digital spaces: an efficient algorithm based on immersion simulations". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6 (1991), pp. 583–598.

[126] E. Vorontsov et al. "Liver lesion segmentation informed by joint liver segmentation". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018, pp. 1332–1335.

[127] U Wallrabe et al. "Micromolded easy-assembly multi fiber connector: RibCon®". In: *Microsystem technologies* 8.2-3 (2002), pp. 83–87.

[128] Tianlu Wang, Kota Yamaguchi, and Vicente Ordonez. "Feedback-prop: Convolutional Neural Network Inference under Partial Evidence". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[129] K. Q. Weinberger and L. K. Saul. "Distance metric learning for large margin nearest neighbor classification". In: *Journal of Machine Learning Research* 10 (2009), pp. 207–244.

[130] Oliver Wirjadi et al. "Characterization of multilayer structures in fiber reinforced polymer employing synchrotron and laboratory X-ray CT". In: *International Journal of Materials Research* 105.7 (2014), pp. 645–654.

[131] Changyan Xiao et al. "A strain energy filter for 3D vessel enhancement with application to pulmonary CT images". In: *Medical image analysis* 15.1 (2011), pp. 112–124.

[132] Changyan Xiao et al. "Multiscale bi-Gaussian filter for adjacent curvilinear structures detection with application to vasculature images". In: *IEEE Transactions on Image Processing* 22.1 (2012), pp. 174–188.

[133] J. Xiao et al. "SUN database: Large-scale scene recognition from abbey to zoo". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 3485–3492. DOI: `10.1109/CVPR.2010.5539970`.

[134] Jianxiong Xiao et al. "SUN Database: Exploring a Large Collection of Scene Categories". In: *International Journal of Computer Vision* 119 (2014), pp. 3–22.

[135] Helmut Zauner et al. "3D image processing for single fibre characterization by means of XCT". In: *Acta Stereologica* (2015).

[136] X. Zhang et al. “A fast segmentation method for high-resolution color images of foreign fibers in cotton”. In: *Composites Part A: Applied Science and Manufacturing* 78.1 (2011), pp. 71–79.

[137] Bolei Zhou et al. “Places: A 10 million Image Database for Scene Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[138] Maciej Zieba et al. “Bingan: Learning compact binary descriptors with a regularized gan”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 3608–3618.