A PIPELINE FOR PRODUCING DEEP LEARNING X-RAY BASED FOREIGN OBJECT DETECTION THROUGH 3D COMPUTED TOMOGRAPHY DATA

Jens van Bijsterveld
s2902702

ABSTRACT

We examine the performance of foreign object segmentation by U-Net through the implementation of the workflow described by *Zeegers et al.* for different amounts of training data. The implemented pipeline computes object radiographs and ground truth foreign object locations for those radiographs by computing the 3D representation of objects. Created training data is used to train U-Net segmentation on this problem space. We conclude that the proposed workflow is valid, but that more research is needed to find the correlation between model performance and amount of training data.

1 INTRODUCTION

1.1 BACKGROUND

These days, many everyday objects are produced in factories. Assembly of large objects like cars, creation of small objects like watches, and even production of edible items like cookies. All these processes have turned into mostly automated processes, requiring as little human intervention as possible.

Automated industrial settings require the invention of high-throughput scanning techniques to assess the quality and safety of created products. For example meat or fish needs to be checked for tiny pieces of glass or bone before they are packaged and shipped to the consumer. In order to do this without destroying the product, some non-invasive cost-effective and fast imaging needs to be performed.

Modern industrial applications of foreign object detection often use X-ray imaging in order to visualize the interior structure of products. The difference in attenuation energy between inserted foreign objects and the base object when looking at the radiograph (2D X-ray scan) of an image is used to automatically determine if the object includes a foreign object (Andriiashen et al., 2021). This same intuition is, for example, used for inspecting your luggage when passing through airport security.

Automating 2D radiograph based foreign object detection proves difficult, since overlapping objects hamper commonly used segmentation methods from finding object boundaries (Sezgin & Sankur, 2004). The segmentation process is even further hampered in a high-throughput industrial setting, since a lower scanning time leads to higher levels of noise in the radiographs. This leads to poor results if the used approach is not extensively tuned for its specific application (Silva et al., 2018).

The recent rise of (supervised) machine learning provides a good chance to create more generalized methods for automated segmentation of foreign objects. Although machine learning methods show improvement over classical segmentation methods, the downside is their reliance on large datasets (Wu et al., 2019). For foreign object detection problems, the creation of such datasets often goes paired with manual labelling of foreign objects, which is a labour intensive and error-prone approach to data collection (Tajbakhsh et al., 2020).

The authors of the paper *A tomographic workflow to enable deep learning for X-ray based foreign object detection* proposed a workflow for automated generation of datasets for training supervised deep learning segmentation methods for foreign object detection in industrial settings (Zeegers et al., 2022). In this paper *Zeegers et al.* propose the usage of 3D computed tomography for segmenting

the foreign objects, since the 3D segmentation task is more straightforward, and therefore able to be computed by global Otsu thresholding (Otsu et al., 1975). This 3D volume is then used to create 2D target segmentations corresponding to the 2D object radiographs, thus automating the creation of 2D training data.

1.2 Problem statement

In this paper, we will evaluate the workflow proposed by *Zeegers et al.* by re-creating the proposed workflow and experimentally testing the quality of its results in a reproducible manner (Zeegers et al., 2022). We will eventually answer the research question **How does the accuracy of deep-learning based foreign object detection using 3D computed tomography data depend on the number of training examples?**

We will answer this research question by varying the amount of 3D (ground truth) objects that are fed into the workflow described by *Zeegers et al.* in order to create training data, and feeding this training data into the well known U-Net deep learning architecture (Ronneberger et al., 2015).

1.3 Overview

The remainder of this paper consists of an overview of the methods used for the experiments (section 2), the results of those experiments (section 3), the conclusions we can derive from them, and notes regarding improvements and possible future work (section 4).

We start with a general overview regarding the workflow proposed by *Zeegers et al.* (subsection 2.1), followed by more in-depth explanations on phantom generation (subsection 2.2), object radiograph generation (subsection 2.3), object reconstruction and following segmentation and target radiograph generation (subsection 2.4), and the deep learning segmentation network (subsection 2.5), concluded with some remarks about the experimental setup (subsection 2.6).

Consequently, the results of these experiments are discussed based on visual inspection (subsection 3.1), model accuracy (subsection 3.2) and average class accuracy (subsection 3.3).

We finish the paper by drawing conclusions from the discussed results (subsection 4.1) and evaluating the methods used in the experiments and possible future work (subsection 4.2).

2 Method

2.1 Method overview

The workflow described by *Zeegers et al.* was implemented as a pipeline capable of creating differing amounts of training data. A visual overview of this pipeline can be seen in Figure 1.

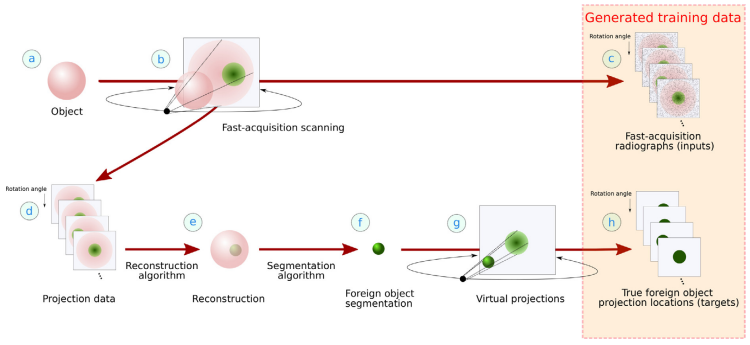


Figure 1: The workflow for creating training data for deep learning based foreign object segmentation as described by *Zeegers et al.* (Zeegers et al., 2022).

As shown in Figure 1, the pipeline starts with generating a specific amount of 3D objects containing foreign objects (phantoms, the amount depends on the amount of required training data). These

phantoms will then be (virtually) scanned. This generated projection data has two purposes. It will be used as input for the deep learning network, and it will be used to reconstruct a 3D representation of the phantom.

The created 3D representation is then used to create 3D representations of the targets (foreign objects) with a classical segmentation method, after which radiographs for the targets are created by once again performing (virtual) scanning. These created radiographs act as the labels (ground truth) for the object radiographs when training the deep learning network, since the created target radiographs correspond to the foreign object locations in the object radiographs (since equal scanning parameters were used).

This pipeline is repeated for differing amounts of objects (which determine the amount of available training data) and its output is used to train a U-Net implementation. This trained network is then evaluated on an independently generated test set consisting of 25 objects.

2.2 Phantom generation

In order to create results we need objects we can feed into the training data generation pipeline. For this paper we opted for the usage of generated 3D objects (phantoms), since this allows for the creation of radiographs without the need to use a physical X-ray scanner.

The phantoms were created in a space of $128 \times 128 \times 128$ voxels. We used the geometrical shape of a $64 \times 64 \times 64$ voxel cube as a base object for creating the phantom. This cube was placed in the middle of the space.

In order to create varying base objects, corners of the cubes were cut off. The cut-off point for each cube corner was determined by a hyperplane defined by three randomly selected points between the midpoint of a cube edge and the corner point. The part of the cube that was opposite the centre of the space with regard to this hyperplane was cut off. A visual representation of this process can be seen in Figure 2.

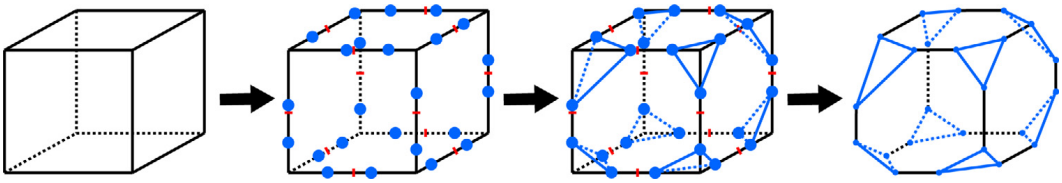


Figure 2: The process of cutting off randomly sized corners from the created cubical phantom. For each of the three edges of each of the eight corners, a random point between the corner point and the halfway point of the edge is determined. All points of a corner define a hyperplane that is used to determine the cube volume that is to be cut off. Figure taken from *Zeegers et al.* (Zeegers et al., 2022). Randomly chosen points are shown in blue. Edge halfway points are shown in red.

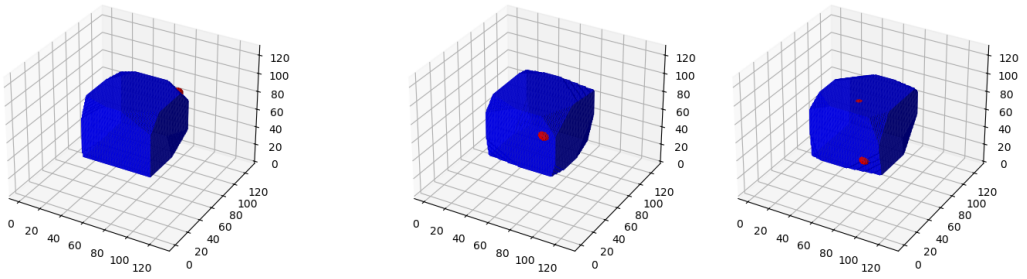
The cube with the corners cut off is defined as the base object. One or two (randomly determined) elliptical 'foreign objects' with a (randomly determined) radius of three to eight voxels are placed either in or on the edge of the base object.

Examples of finalized phantoms can be seen in Figure 3.

2.3 Object radiograph generation

The workflow as defined by *Zeegers et al.* uses the FleX-ray CT system to create radiographs for (approximate) cubes of modelling clay with inserted gravel (Coban et al., 2020) (Zeegers et al., 2022). In order to stay true to the pipeline, the parameters for our virtual scanning of the created phantoms (subsection 2.2) uses the same parameters as the scanning in the FleX-ray CT scanner. An overview of these parameters can be found in Table 1.

In the original workflow, high quality radiographs are made by using 1800 projections angles over a 360-degree rotation. The effect of an industrial setting is induced by using a low exposure time (20ms), which produces sufficient noise in the object reconstructions. Since we are using virtual



(a) A phantom containing a single foreign object.

(b) A phantom containing a single foreign object.

(c) A phantom containing two foreign objects.

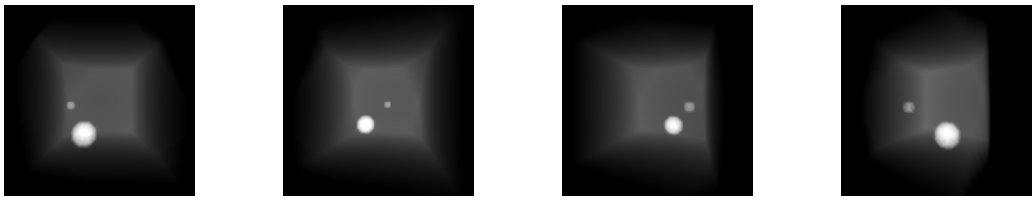
Figure 3: Three examples of created phantoms for usage in the training data generation pipeline.

Table 1: FleX-ray parameters as used in the original workflow. These same parameters were used for the virtual projections performed in the experiments.

Beam description	Cone
Distance source-object	44.14 mm
Distance source-detector	69.80 mm
Detector rows	128
Detector columns	128
Detector pixel size	1.05 mm

projections, we cannot alter the exposure time. In order to emulate the speed of an industrial setting, we use 180 projections over a 360-degree angle in the creation of our scans.

The resulting radiographs are saved for each projection angle. Examples of resulting (contrast enhanced) radiograph images can be seen in Figure 4.



(a) A radiograph of a phantom at 0 degrees.

(b) A radiograph of a phantom at 100 degrees.

(c) A radiograph of a phantom at 200 degrees.

(d) A radiograph of a phantom at 300 degrees.

Figure 4: Four examples of radiographs from a phantom with two foreign objects from different projection angles.

2.4 Target radiograph generation

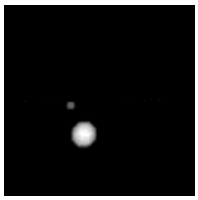
From the radiographs generated by scanning the phantoms, reconstructions are made by using the SIRT (Simultaneous Iterative Reconstruction Technique) algorithm (Gregor & Benson, 2008). The SIRT algorithm is used for 100 iterations.

After reconstruction, the foreign object is determined by 3D segmentation using global Otsu thresholding (Otsu et al., 1975). This thresholding technique maximizes the inter-class variability between the background and foreground in order to segment an object.

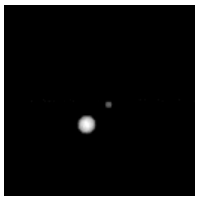
First, the entire object (cube and foreign objects) is segmented from the background. Then the background is set to the average intensity value of the cube and its foreign objects, and the new image is segmented again by using Otsu thresholding to find the foreign object locations.

The 3D object containing only the foreign objects is (once again) virtually scanned with similar parameters to the main object scanning procedure (subsection 2.3) in order to create radiographs for the targets. Since we are using equal parameters to the previous scans, the created target radiographs match the created object radiographs with regard to foreign object locations.

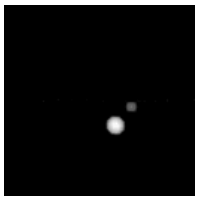
Examples of created target radiographs can be seen in Figure 5.



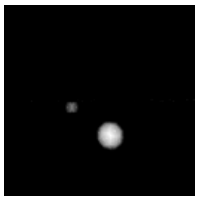
(a) A target radiograph of a phantom at 0 degrees.



(b) A target radiograph of a phantom at 100 degrees.



(c) A target radiograph of a phantom at 200 degrees.



(d) A target radiograph of a phantom at 300 degrees.

Figure 5: Four examples of radiographs from a phantom with two foreign objects from different projection angles.

2.5 Deep segmentation network

For the deep learning segmentation model architecture, we used the U-Net architecture, which is a well known architecture for segmentation of biomedical images (Ronneberger et al., 2015). U-Net was chosen because it was designed to use available training data more efficiently, meaning it should be capable of good performance with small amounts of training objects.

The U-Net architecture consists of blocks of convolutional layers (encoder blocks) and blocks of transposed convolutional layers (decoder blocks). Dropout and pooling operations are added to increase the stability of the network. Skip connections are added to decrease the amount of information loss.

The encoder blocks increase the amount of channels in the image while decreasing the image size, and the decoder blocks decrease the amount of channels in the image while increasing the image size. The final layer creates an image with as many channels as classification classes (two for our segmentation problem), where each channel contains the probability of that pixel being that class.

An overview of the originally proposed U-Net architecture can be seen in Figure 6.

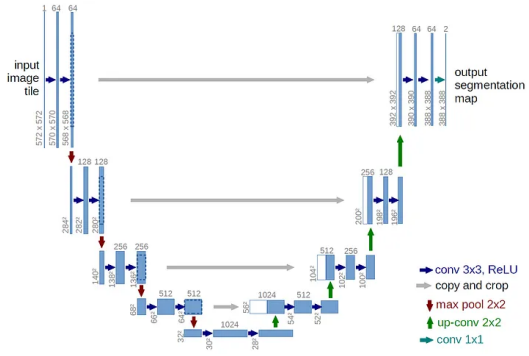


Figure 6: The U-Net network architecture. U-Net is called U-Net because of the shape of its architecture.

Our U-Net implementation was trained with the object radiographs as input (converted to RGB representation in order to create three channels) and the target radiographs as output. In the workflow by *Zeegers et al.* the input radiographs are scaled to 128 $\times$ 128 in order to save computation time. Since the radiographs used for the experiments were already of size 128 $\times$ 128, no rescaling was needed.

The results of the trained U-Net network for different amounts of training objects can be seen in subsection 3.1.

2.6 EXPERIMENTAL SETUP

The code for the experiments was written in python (Van Rossum & Drake, 2009). Most basic operations were implemented using the numpy library (Harris et al., 2020). The ASTRA toolbox was used for performing virtual projection and reconstruction through the SIRT algorithm (van Aarle et al., 2016) (Gregor & Benson, 2008). Diplib was used to perform image operations like Otsu thresholding (DIP) (Otsu et al., 1975). Tensorflow was used for the U-Net implementation (Abadi et al., 2015).

Since the 3D implementations of the projection and scanning algorithms were used, GPU computing was mandatory. During the experiments, all intermediate data was saved as either numpy file objects (`.npy`) or `.tiff` files in order to preserve progress if an unforeseen error was encountered. The experiments thus required a large amount of file storage and a large amount of computing time, even while using a very limited amount of objects / projections.

In order to evaluate U-Net segmentation performance, visual assessment, the tensorflow accuracy measurement and image-based average class accuracy were used (Grandini et al., 2020). The equation that was used for computing the average class accuracy (also known as balanced accuracy) can be seen in Equation 1, where TP stands for true positive class prediction, FP stands for false positive class prediction, obj stands for the foreign object class, and bg stands for the background class.

$$ACA = \frac{1}{2} \cdot \left(\frac{TP_{obj}}{TP_{obj} + FP_{bg}} + \frac{TP_{bg}}{TP_{bg} + TP_{obj}}\right) \tag{1}$$

The code for the experiments can be found in this paper's GitHub repository (van Bijsterveld, 2024). Projection and reconstruction code was inspired by example code by *Tom Roelandts* (Roelandts, 2024). The U-Net implementation code was inspired by the GitHub repository by *Vidushi Bhatia* (Bhatia, 2024).

3 RESULTS

In order to evaluate the effect of the amount of training samples on U-Net performance for foreign object segmentation, the pipeline described in section 2 was executed for 1 up to 100 phantoms. Eventually, U-Net segmentation models were trained for training sets of sizes 1, 3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90 and 100 phantoms, where each phantom corresponds to 162 training examples and 18 validation examples, since 180 radiographs were produced per phantom and a 90/10 train/validation split was used.

Segmentation networks were trained for 20 epochs and then evaluated on an independent test set consisting of 25 phantoms. The ADAM optimizer with a learning rate of 0.001 and sparse categorical cross-entropy was used for training.

3.1 MANUAL INSPECTION

Side by side visualizations of the input, target and prediction were made for each model, in order to visually inspect its results. These visualizations can be seen if Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11 for the networks trained with 1, 5, 10, 40 and 100 training objects, respectively.

We observe an increase in correct foreign object segmentation in Figures 7, 8 and 9 (for 1, 5 and 10 training objects). From then on out (for 40 and 100 training objects, Figure 10 and Figure 11), we

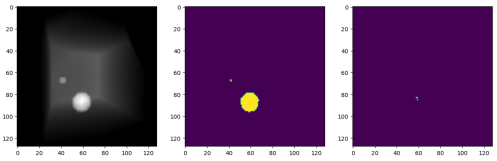


Figure 7: Visual representation of the output of the U-Net segmentation model trained with 1 training object. The leftmost figure shows the object radiograph (input). The middle figure shows the foreign object radiograph (target). The rightmost figure shows the segmentation model's prediction after 20 epochs of training.

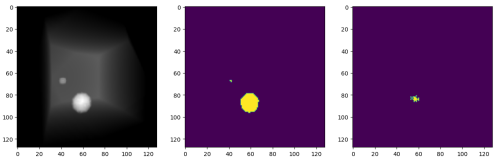


Figure 8: Visual representation of the output of the U-Net segmentation model trained with 5 training objects. The leftmost figure shows the object radiograph (input). The middle figure shows the foreign object radiograph (target). The rightmost figure shows the segmentation model's prediction after 20 epochs of training.

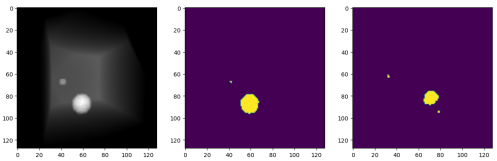


Figure 9: Visual representation of the output of the U-Net segmentation model trained with 10 training objects. The leftmost figure shows the object radiograph (input). The middle figure shows the foreign object radiograph (target). The rightmost figure shows the segmentation model's prediction after 20 epochs of training.

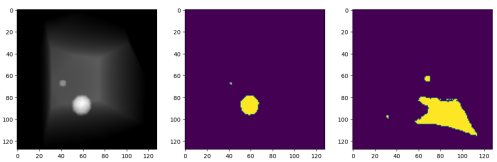


Figure 10: Visual representation of the output of the U-Net segmentation model trained with 40 training objects. The leftmost figure shows the object radiograph (input). The middle figure shows the foreign object radiograph (target). The rightmost figure shows the segmentation model's prediction after 20 epochs of training.

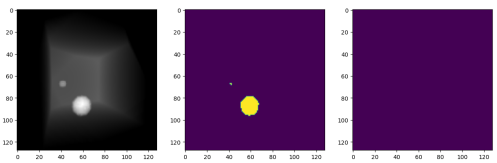
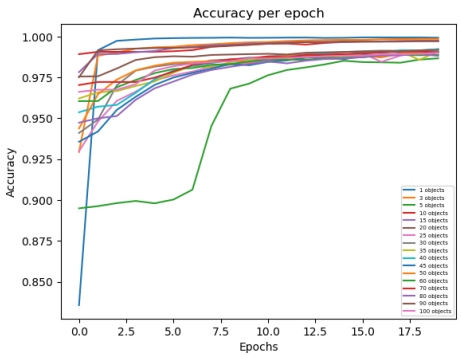


Figure 11: Visual representation of the output of the U-Net segmentation model trained with 100 training objects. The leftmost figure shows the object radiograph (input). The middle figure shows the foreign object radiograph (target). The rightmost figure shows the segmentation model's prediction after 20 epochs of training.

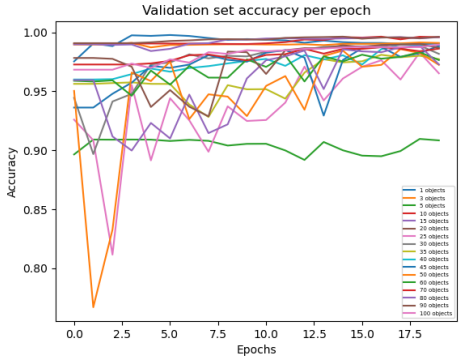
see a decrease in model accuracy, either through bad segmentation (Figure 10) or overgeneralization (Figure 11).

3.2 Accuracy

During training of the segmentation networks, the accuracy (percentage of correctly classified image elements) was calculated for each training epoch. The accuracy was calculated for both the training set and a validation set. The results for accuracy during training can be seen in Figure 12.



(a) Accuracy during training for the training set.



(b) Accuracy during training for the validation set.

Figure 12: Model accuracy during model training for the training set and the validation set. Note that the size (and variability) of the validation set scales with the size (and variability) of the training set.

For the training set, we observe a standard learning curve for most segmentation models. Most models learn quickly and then plateau around 97.5 percent accuracy. The model trained with 5 training objects learns more slowly, and reaches approximately 95.0 percent accuracy.

For the validation set we see a lot more variability for the accuracy during training. Most models eventually reach a stable state of around 97.5 percent accuracy, while the model trained with 5 training objects reaches an accuracy of approximately 90.0 percent.

3.3 Average class accuracy

Since the accuracy is not the best measure of model performance for this task (most of the image is not a target, meaning high accuracy can be achieved by predicting the background class for all image elements, see Figure 11), model performance was further evaluated by computing the average class accuracy (subsection 2.6) (Grandini et al., 2020).

The average class accuracy was calculated for each test set element, and averaged for each model. The results can be seen in Figure 13.

We observe quite low values for the average class accuracy (ACA). We can see that the ACA values are mostly around 0.5, which corresponds to an accuracy that is equivalent to random chance ($\frac{1}{2}$ for a two class classification problem). We see some higher values for ACA, but those come paired with an increase in the standard deviation.

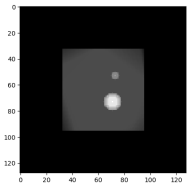
4 Discussion and conclusions

4.1 Conclusion

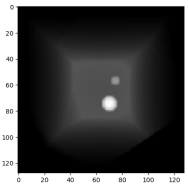
In this paper, we examined the correlation between the amount of training objects and deep learning based segmentation model performance for the workflow proposed by *Zeegers et al.* (Zeegers et al., 2022).

Figure 13: The average class accuracy plotted against the number of training objects. Average class accuracy is represented as a blue line. The standard deviation of the class accuracy is shown as a shaded region.

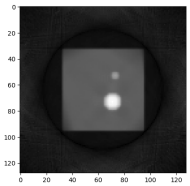
The workflow from the paper by *Zeegers et al.* was implemented and performs as expected. It creates accurate datasets for training deep learning based segmentation models without human intervention. An overview of a single phantom going through this pipeline can bee seen in Figure 14.



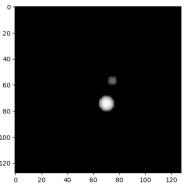
(a) A phantom is created. 2D representation of a 3D phantom.



(b) Radiographs of the phantom are computed. Radiograph of the phantom at zero degrees.



(c) Using the radiographs from b, the phantom is reconstructed. 2D representation of a 3D phantom.



(d) 4. The reconstructed phantom is segmented and projections are made from the segmented foreign objects. Radiograph of the segmented phantom at zero degrees.

Figure 14: A phantom going through the pipeline. Results b and d are used as training data for a deep learning segmentation network.

The expected result for model performance when looking at the original paper would be an increase of model accuracy (measured through average class accuracy) together with an increase in amount of training objects, plateauing around 60 training objects. This is however not the case for the results produced by our experiments (section 3). Although visual inspection of the first few trained deep segmentation networks seemed promising, model performance quickly deteriorated afterwards.

In our experiments, we observe that the quality of U-Net segmentation does not depend on the amount of training objects, since U-Net segmentation of foreign objects fails regardless of the amount of training data. High accuracy can be caused by the significant class imbalance between the background class (the background and main object) and the foreground class (the foreign objects), while the actual accuracy (measured by, for example, average class accuracy) remains low.

We conclude that, based on the research done for this paper, no distinct correlation can be found between the accuracy of deep learning based foreign object detection using 3D computed tomography data and the amount of training examples. Further research will be needed to find this correlation.

4.2 Discussion

Although the implemented pipeline for unsupervised dataset creation worked perfectly, deep learning based segmentation model performance was lacking. There might be a multitude of reasons for this.

First of all, *Zeegers et al.* created a different amount of projections for each phantom (1800 instead of 180, corresponding to a radiograph each 0.2 degrees instead of a radiograph each 2 degrees). This might have increased model performance by increasing the amount of training data for each model by a factor 10.

Second of all, it is unclear how many training epochs *Zeegers et al.* used while training their models. The authors only state that each model was trained for 9 hours, but do not mention the amount of training epochs. Due to the limited computing power and storage space available for this paper, the number of epochs was set at 20.

An additional experiment was done by increasing the amount of epochs to 200 for the model trained with 5 phantoms. According to the paper by *Zeegers et al.*, this model would result in an ACA of approximately 85 percent. Unfortunately, after training for approximately 7 hours, our model reached an ACA of approximately 49.97 percent, which once again corresponds to random chance. The accuracy during training for this model can be seen in Figure 15

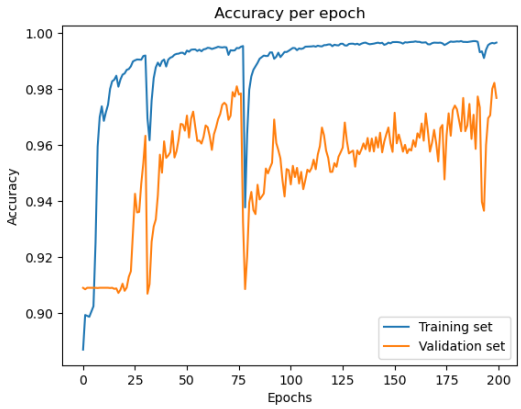


Figure 15: Model accuracy during model training for the training set and the validation set for a U-Net model trained with five phantoms.

Finally, the paper by *Zeegers et al.* does not mention their choice of hyperparameters or loss function. Extensive hyperparemeter tuning or a different choice of loss function could improve model performance, but was outside the scope of this project due to the limited amount of available computing power.

Further research on this subject can be done by reproducing the experiments described in this paper with a larger amount of computing time / power and a larger amount of storage space in order to find a correlation between the amount of training objects and segmentation network performance.

Further extension on this research would be to investigate model performance for segmentation into three classes (background, object and foreign object) instead of two, testing pipeline and model performance with added Poisson noise, testing the impact of normalization through usage of darkfield and/or flatfield images, or the trying of different deep learning segmentation model architectures, such as the modifications to U-Net as proposed by *Zeegers et al.*, or the usage of a fully different network architecture such as, for example, Mixed-Scale Dense (MSD) (Pelt & Sethian, 2018).

REFERENCES

DIPlib: A Scientific Image Processing Library. https://diplib.org/. Accessed: June 6, 2024.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Vladyslav Andriiashen, Robert van Liere, Tristan van Leeuwen, and Kees Joost Batenburg. Unsupervised foreign object detection based on dual-energy absorptiometry in the food industry. *Journal of Imaging*, 7(7):104, 2021.

Vidushi Bhatia. U-net implementation. https://github.com/VidushiBhatia/U-Net-Implementation/tree/main, 2024. Accessed: June 7, 2024.

Sophia Bethany Coban, Felix Lucka, Willem Jan Palenstijn, Denis Van Loo, and Kees Joost Batenburg. Explorative imaging and its implementation at the flex-ray laboratory. *Journal of Imaging*, 6(4):18, 2020.

Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.

Jens Gregor and Thomas Benson. Computational analysis and improvement of sirt. *IEEE transactions on medical imaging*, 27(7):918–924, 2008.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11 (285-296):23–27, 1975.

Daniël M Pelt and James A Sethian. A mixed-scale dense convolutional neural network for image analysis. *Proceedings of the National Academy of Sciences*, 115(2):254–259, 2018.

Tom Roelandts. Tom roelandts' website. https://tomroelandts.com/, 2024. Accessed: June 7, 2024.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

Mehmet Sezgin and Bu¨ lent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–168, 2004.

Gil Silva, Luciano Oliveira, and Matheus Pithon. Automatic segmenting teeth in x-ray images: Trends, a novel data set, benchmarking and future perspectives. *Expert Systems with Applications*, 107:15–31, 2018.

Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N Chiang, Zhihao Wu, and Xiaowei Ding. Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. *Medical image analysis*, 63:101693, 2020.

Wim van Aarle, Willem Jan Palenstijn, Jeroen Cant, Eline Janssens, Folkert Bleichrodt, Andrei Dabravolski, Jan De Beenhouwer, K. Joost Batenburg, and Jan Sijbers. Fast and flexible x-ray tomography using the astra toolbox. *Opt. Express*, 24(22):25129–25147, Oct 2016. doi: 10.1364/OE.24.025129. URL https://opg.optica.org/oe/abstract.cfm?URI=oe-24-22-25129.

Jens van Bijsterveld. foreign-object-pipeline. https://github.com/JensVanBijs/foreign-object-pipeline/tree/master, 2024. Accessed: June 7, 2024.

Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

Hao Wu, Qi Liu, and Xiaodong Liu. A review on deep learning approaches to image classification and object segmentation. *Computers, Materials & Continua*, 60(2), 2019.

Mathé T. Zeegers, Tristan van Leeuwen, Daniël M. Pelt, Sophia Bethany Coban, Robert van Liere, and Kees Joost Batenburg. A tomographic workflow to enable deep learning for x-ray based foreign object detection. *Expert Systems with Applications*, 206:117768, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.117768. URL https://www.sciencedirect.com/science/article/pii/S0957417422010429.