DOI: 10.58895/ksp/1000150865 -14 erschienen in:
Forum Bildverarbeitung 2022 | Image Pocessing Forum 2022
DOI: 10.5445/KSP/1000150865 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000150865/

Deep learning and active learning based semantic segmentation of 3D CT data

Markus Michen[1] and Ulf Haßler[1]

Fraunhofer-Entwicklungszentrum Röntgentechnik EZRT,
Flugplatzstr. 75, 90768 Fürth

Abstract In this paper, we developed a tool that uses active learning and deep learning together for segmentation of 3D CT data. We demonstrate the results of the method using the use case of plant segmentation. In addition, we compare the method with a baseline and a classical image processing-based algorithm.

Keywords Deep learning, active learning, semantic segmentation, plant segmentation, image processing, u-net

1 Introduction

Automated segmentation of 3D CT data is a vast field of application. Especially in the medical environment, there is currently a transition from conventional methods based on classical image processing to Machine Learning / Deep Learning (ML/DL) based methods [1,2]. Much of the aforementioned success of Deep Learning is due to the large number of publicly available annotated datasets, for example, the ImageNet database [3]. One of the major challenges is the necessity to acquire sufficient ground truth data for modeling. However, this data are usually not available in sufficient quantities, especially for industrial use cases. Moreover, the annotation of this data turns out to be an extremely time-consuming and very expensive task, especially for large 3D datasets.

Thus, we need effective methods to reduce the labeling effort. One such method is active learning, a collection of techniques that support machine learning algorithms to achieve better results with less labeled

training data. The learning algorithm can interactively prompt a user to assign the correct labels to new data points. To do this, the algorithm should ask questions that promise a high information gain in order to keep the number of questions as small as possible.

These questions, called queries, can be grouped into three main types: stream-based selective sampling, membership query synthesis, and pool-based sampling. Stream-based selective sampling assumes a stream of incoming unlabeled data points x. The current model and a measure of informativeness measure $I(x)$ are used to decide for each incoming data point whether to ask the oracle for an annotation. In membership query synthesis, the data points are not drawn, but rather the model generates new data points in a way that it considers informative to itself. With pool-based sampling, a batch b is selected from the unlabeled dataset. The current model is used to predict the sample stack and obtain a measure of informativeness $I(b)$. Based on this measure, the best N samples are selected to be annotated by the oracle [4].

Overall, Deep Learning has strong capabilities in processing data through automatic feature extraction, but requires a very large amount of annotated data to do so. Active Learning, on the other hand, has the potential to effectively reduce the effort required for labeling. The combination of deep learning and active learning support each other, so their application potential improves significantly. Therefore, we have developed a tool that allows us to apply active learning to the area deep learning segmentation of 3D CT data.

We demonstrate the use of our tooling on the basis of plant segmentation, as plant breeding has undergone rapid progress in recent decades. In this context, targeted plant breeding, for example of climate-resistant strains, is also becoming increasingly important [5]. Innovative analysis methods, such as 3D segmentation, play an essential role in this context, enabling seedlings and seeds to be assessed qualitatively.

The segmentation task here is to divide the 3D CT scan of the plant inside a container in folded paper into the classes plant, paper and background (see figure 1). Through use of the segmentation the individual plants can be evaluated and classified by downstream applications later. It is particularly difficult to distinguish the seedlings from the paper. Paper and seedling absorb X-rays to a similar degree, so there is virtually no usable contrast difference that could be used for

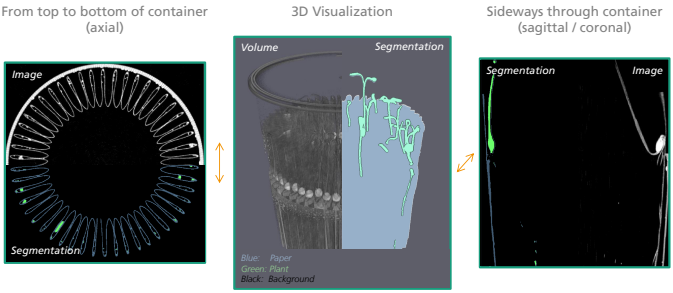

Figure 1: Data for the plant segmentation. The data is split in raw image or volume and its corresponding segmentation. On the left the volume is sliced in axial direction. In the middle a 3D rendering can be seen. And on the right the sagittal/coronal direction is shown.

segmentation. This is also affected by the limited resolution of only 140 µm and noise, which is why incorrect segmentations can easily occur. Either components of the seedling are assigned to the paper or vice versa. This hinders the subsequent assessment of the seedling in the downstream application due to incorrectly calculated characteristics.

2 Methods

Our method operates in three main phases (see figure 2). In the pre-training phase, an initial network (currently 3D U-Net) is trained from weak labels. These can be derived from existing classical image processing pipelines, simulations or rough hand-annotations.

Subsequently, this pre-trained network is passed to the active learning phase. The active learning phase itself also consists of several steps, namely inference, location, visualization/interaction, and training. During inference, the segmentation network generates a segmentation map, which is then analyzed during the location phase. Then the user can visualize the results and interact with them to correct invalid segmentations. Next, the areas corrected by the user are retrained during the training phase and the weights of the segmentation network are updated. A graphical user interface guides the user through these four steps until a visually satisfactory result is achieved or an application-specific condition is met.

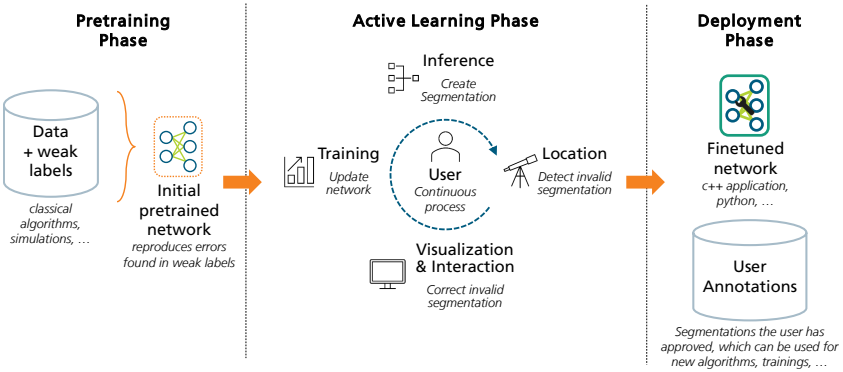

Figure 2: Overall conceptual process of the developed deep learning and active learning approach.

Finally, the resulting fine-tuned network can be deployed. As an additional result, all corrections made by the user during the active learning phase can be used for future algorithms or training.

2.1 Pre-training phase and network architecture

In the pre-training phase, the network is initially trained in such a way that later, in the active learning phase, the segmentation is almost correct and only invalid segmentations have to be corrected and re-trained. For this, already existing classical algorithms (based on thresholding, filtering, . . .) or simulations can be used as weak labels.

The U-Net architecture used consists of a simple 3D U-Net (see figure 3). It is 5 levels deep with 2 convolution blocks per level. With each level the number of feature maps doubles and the spatial resolution halves. The convolution blocks consist of 2 convolution layers with batch normalization [6], swish activation [7] and a residual connection. In the decoding path the feature maps are upsampled and concatenated by simple upsampling. The last layer is $1x1x1$ convolution with Softmax activation and represents our final segmentation. The entire 3D input volume is usually too large to be processed at once, so it is processed block by block through subvolumes of size 64^3.

The training parameters are set as follows. As loss function we

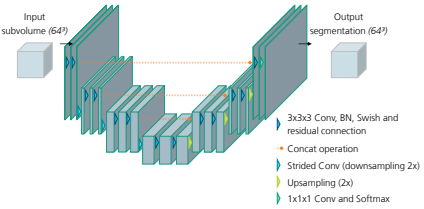

Figure 3: Schematic of the employed u-net network architecture.

choose the sum of dice and cross entropy loss, as in [8]. For the optimizer, we use Adam [9] with a learning rate of $3e^{-4}$ and cosine decay. As regularization, we set a weight decay of $2e^{-5}$ [10] and also use label smoothing of 0.1. Additionally, we use augmentations to increase the training data. We use contrast, noise, affine transformations, flips, blur and artifacts augmentations with varying strength. The network is implemented using TensorFlow [11] and the augmentation pipeline makes use of the TorchIO package [12]. The training has been conducted using a NVIDIA GTX Titan X with 12 GB of GPU RAM.

2.2 Active learning phase

After the network has completed the pre-training phase and has reached suitable convergence, it is passed on to the active learning phase. Here the user is in the focus, and first he is presented with the following view within the simple application we developed, with which he can interact with the current dataset. In figure 4 three orthogonal sliceable views with which the dataset can be navigated can be seen and the toolarea in which multiple tools are available for the user. The user has access to brush, image processing operations (flood-fill, morphology, clustering, ...), 3D visualization, neural network training and use-case specific functions.

In general, 4 sub-steps are then performed within the active learning phase: namely, inference, location, visualization/interaction, and training.

Inference. Here the network prediction is executed. In the field of 3D CT data segmentation, the volumes are often large, with sizes of several GB or more, which prevents the direct use of a segmentation network

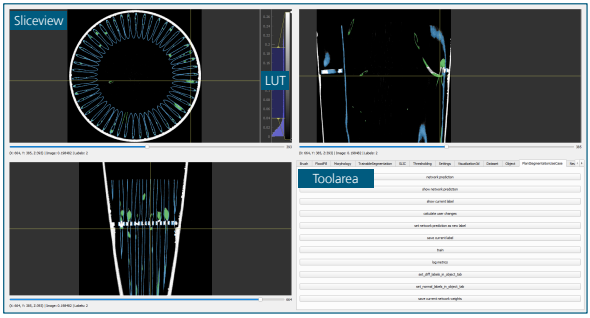

Figure 4: Overview of the graphical user interface guides the user through the active learning phase. In the top right, top left and lower left, orthogonal sliceable views can be seen, that allow the user to navigate the through the data and overlayed segmentation. In the bottom right the toolarea can be seen.

due to the limited GPU RAM. Therefore, to perform segmentation with such volumes, we need to split them into smaller blocks (usually 64^3). Each of these blocks is then segmented individually by the network. In addition, overlapping is performed at the edges of the blocks to compensate for the lack of spatial information at the edges. Finally, all the blocks are merged to form the total volume.

Location. In the localization phase, the user has to find and correct incorrect segmentations. Since this is a very time-consuming process to do manually, we have developed a way to quickly and semi-automatically present potentially incorrect areas to the user. To do this, we use a random forest that classifies the objects contained in the current segmentation. It is trained by the user on the basis of a few examples. For this purpose, first the current segmentation is analyzed by a connected component analysis (CCA). Then, features are calculated for each of the connected objects (e.g. size, mean, eigenvalues, ...). Now the user has to label at least one object of each desired class (for example: paper, seedling, faulty, ...). After that, the random forest can be trained and applied to all contained objects. The user is then shown the objects that have been classified and can then improve their segmentation. The GUI and the random forest pipeline are shown in figure 5.

Visualization/Interaction. After a wrong segmentation has been

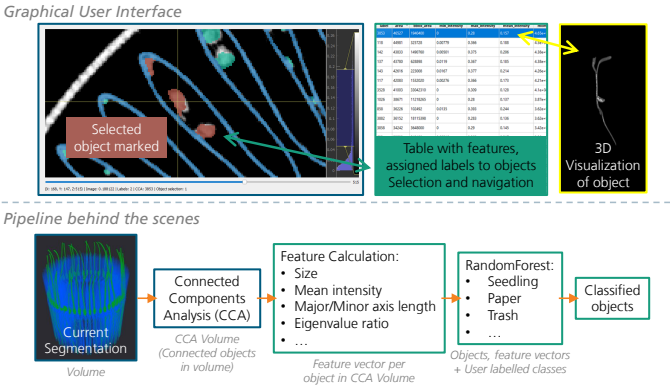

Figure 5: Location phase overview. On the top the GUI part of the location phase can be seen. In the middle the table containing features of the objects can be selected. On the left you can see a cutout of the data and the object is slightly highlighted in red. On the right a 3D visualization of the object is displayed. The lower part shows the pipeline running in the background. The gray text describes the data flow from plain voxels to connected objects and their features.

found in the localization phase, the user has to correct it. This is done with the help of the three orthogonal views in the GUI and the available tools. Most of the time, the corrections that need to be made are small local corrections, such as roots that are incorrectly marked as paper. However, painting pixels is difficult and painting voxels turns out to be even more difficult. That's why we provided the brush tool with special modes for segmenting plants. After all, the brush tool is the most commonly used tool for local segmentation changes. It should be easy (and fun) to use and support many automatic modes so that the user can segment as many voxels as possible by hand with as little effort as possible. In figure 6 the brush usage of the brush tool is shown along with its special Frangi [13] filter mode.

Training. After the localization, visualization and interaction phases mentioned above, the training phase can begin. The goal of our active learning process is that the user annotates as little as possible, but as much as necessary to correct the wrong segmentations. Therefore, the changes in the loss of the network are also rather small, which could hinder the learning of the corrected regions. To compensate for this

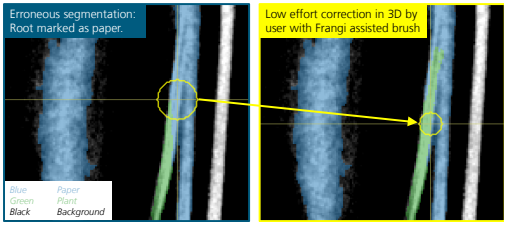

Figure 6: Example of an incorrect segmentation and its correction. The Frangi filter can select tubular structures, which makes it easier to separate them from the planar paper, allowing the user to correct the incorrect segmentation more easily than by manually tracing each voxel.

imbalance, voxel-wise loss weighting is used to force learning of the regions corrected by the user. The weight calculation is similar to scikit-learns class weights function [14]. The training parameters are the same as in the pretraining phase mentioned above.

Iteration. Finally, the figure 7 shows an iteration of the active learning process of the developed tool. Starting in the inference phase, the current network generates a segmentation. Then, in the localization phase, the incorrect region is found and presented to the user. Subsequently, the user corrects the incorrectly segmented voxels. After finishing training with the new annotations, the next iteration can start. In the upper right of the figure 7, the result after the iteration can be seen on another area that was not annotated by the user.

2.3 Deployment phase

After the active learning phase has been completed, the resulting fine-tuned network can be passed on to the deployment phase. Here, it is then used for inference in another application. In the case of plant segmentation, the output of the network is used to analyze individual seedlings and their characteristics for subsequent seed selection and breeding.

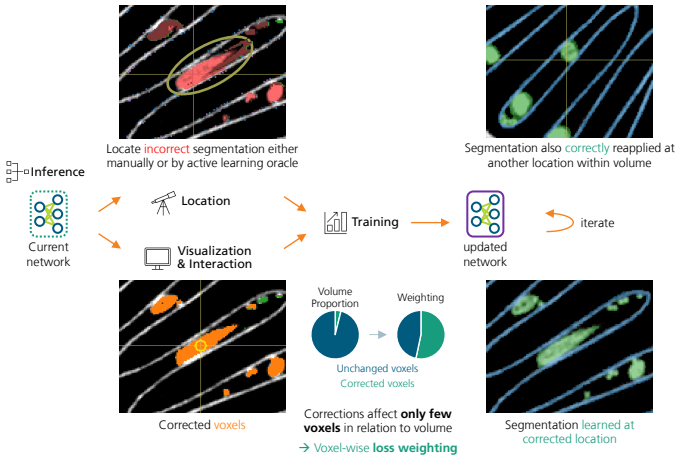

Figure 7: Overview of the usual active learning workflow with example segmentations in the different steps. In the top right the result after the iteration is shown.

3 Results

We evaluated our developed tool on the use case of segmentation of 3D CT scans of plants. The seedlings grow in a plastic container in folded paper. Due to the similar attenuation coefficients, it is particularly difficult to distinguish plant and paper. We compare the performance of the pre-trained network and the fine-tuned network with the performance of a classical image processing-based algorithm [5]. The methods are compared visually by inspection and by calculating segmentation metrics. To give no algorithm an advantage, we manually created a ground truth scan from the test set from scratch without using algorithmic assistance. In order not to let the effort explode, we evenly distributed two slices from each of the three orthogonal directions (see figure 1) for annotation. Each of these six slices took the annotator an average of 20 minutes, extrapolating to the total scan size of about 800^3, this would require about 16 days for the entire scan in the worst case, which would be impractical.

Figure 8 show the comparison of the segmentation with the respective ground truth slice from the two different directions. As can

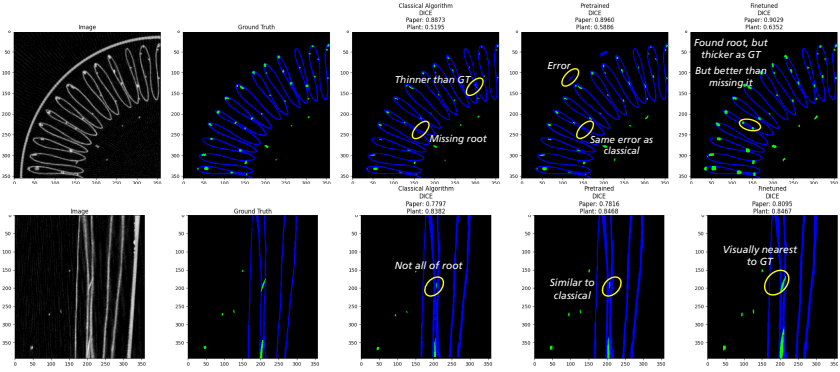

Figure 8: Results for the ground truth slices of different directions (top: axial, bottom: sagittal/coronal). The method names and their dice results are shown above the image. Various points of interest are highlighted by yellow ellipses.

be seen, the classical algorithm generally segments the roots more sparsely than the ground truth. In some cases, the roots are completely missed, which is a fatal error for the final application in the deployment phase. The pre-trained network reproduces the errors of the classical algorithm, which is to be expected after it has been trained with data from the classical algorithm. The fine-tuned network finds roots missed by the other two methods, but segments them a bit too thick. Nevertheless, such an error is not as serious as missing roots.

The figure 9 shows the metrics of the different methods. It can be seen that all metrics are quite close to each other. The classical algorithm can only convince in one metric, while the pre-trained network achieves the highest score in 2 out of 12 cases. In the remaining 9 out of 12 cases, the fine-tuned network achieves the highest scores. This is also in agreement with the assessment in the visual inspection.

4 Conclusion

Overall, the results achieved with our active learning tooling in plant segmentation are very promising. Although all metrics are quite close to each other, we have a performance gain of about 1% in terms of the DICE score. Furthermore, qualitatively visually, the DL segmentation

Metric	Accuracy			Area under curve			Dice			Intersection over Union (IoU)		
Algorithm	Background	Paper	Plant	Background	Paper	Plant	Background	Paper	Plant	Background	Paper	Plant
1_classical	0.9827	0.9825	**0.9962**	0.9516	0.9495	0.8499	0.9906	0.8708	0.7456	0.9815	0.7735	0.6064
2_pretrained	0.9825	0.9827	0.9958	0.9630	0.9572	0.9019	0.9905	0.8741	**0.7583**	0.9812	0.7789	**0.6178**
3_finetuned	**0.9837**	**0.9848**	0.9949	**0.9788**	**0.9682**	**0.9636**	**0.9911**	**0.8894**	0.7545	**0.9824**	**0.8031**	0.6109

Figure 9: Table of the calculated segmentation metrics. In the top row, the metric can be read. In the second row, the class to which it refers. The last three rows show the results of the individual algorithms. The metric of the best method is highlighted in green.

results are ahead. Additionally, we did not use any prior knowledge about scan geometry, container, paper and plant type. This makes the DL approach much easier to adapt. In the future, other active learning approaches or new network architectures can be integrated to further increase the performance.

Acknowledgments

This work was supported by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport and Technology through the Center for Analytics Data Applications (ADA-Center) within the framework of "BAYERN DIGITAL II".

References

1. O. Ronneberger, P. Fischer, and T. 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.
2. Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2016, pp. 424–432.
3. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
4. B. Settles, "Active learning literature survey," 2009.

5. M. Rehak, U. Haßler, T. Grulich, N. Wörlein, F. Porsch, I. Götz, and A. Wolff, "Der phenotest–ein automatisiertes ct-system zur phänotypisierung von zuckerrübenkeimlingen," in *Forum Bildverarbeitung 2018*. KIT Scientific Publishing, 2018, pp. 217–228.
6. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
7. S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, 2018.
8. F. Isensee, J. Petersen, A. Klein, D. Zimmerer, P. F. Jaeger, S. Kohl, J. Wasserthal, G. Koehler, T. Norajitra, S. Wirkert *et al.*, "nnu-net: Self-adapting framework for u-net-based medical image segmentation," *arXiv preprint arXiv:1809.10486*, 2018.
9. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
10. I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
11. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/
12. F. Pérez-García, R. Sparks, and S. Ourselin, "Torchio: a python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning," *Computer Methods and Programs in Biomedicine*, p. 106236, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260721003102
13. A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever, "Multiscale vessel enhancement filtering," in *International conference on medical image computing and computer-assisted intervention*. Springer, 1998, pp. 130–137.
14. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn:

Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.