Bonn-Aachen International Center for Information Technology (B-IT)

University of Bonn

Master Programme in Life Science Informatics

Master Thesis

Segmentation of Plant Root MRI Images With 3D U-Net

Author:
Yi Zhao

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Prof. Dr. Thomas Schultz

Advisor:
Nils Wandel

Submitted: 12.11.2019

Abstract

To study the mechanism of plant root water uptake, Magnetic Resonance Imaging (MRI) can be used to observe the roots while they grow in the opaque soil. However, the low signal-to-noise ratio and resolution of the 3D MRI images hinder the extraction of structural models from them. To deal with this, we use a 3D convolutional neural network to segment MRI images into root and non-root while increasing the resolution of the output.

The structure of our network is designed based on the 3D U-Net, a widely used segmentation method. Our network takes a 3D image crop as the input, and the segmentation outputs are assembled as the result for the whole image. For improving a previous dataset of synthetic noisy root images, we added randomly generated roots combined with real soil images. The network is trained and validated on this new dataset and evaluated on real MRI images with a special F1 metric that is robust against misalignments between the image and its annotation.

Several experiments are done to improve the segmentation performance. We experimented with additional input channels, including the root image at a later time point and two types of location-dependent information. To remove the part of the learning task that is unnecessarily hard, we tried making the network ignore the root-soil border region during training. Besides, we tested a loss function that weights the root voxels more, to overcome the imbalance between the root and non-root voxels. Moreover, for faster learning convergence, we tried sampling image crops that have more root voxels.

In the end, the structural models extracted from our segmentation results reconstruct the real roots successfully, except a small number of larger gaps in the roots and some minor false positives.

Contents

List of Figures

List of Tables

1 Introduction

The mechanism of root water uptake is critical in understanding plant growth. In order to study that, one approach is to analyze root growth patterns under different soil watering conditions. For this, non-invasive 3D imaging methods are required, to observe the roots while they grow in the opaque soil. Such methods include X-Ray CT, neutron radiography and magnetic resonance imaging (MRI). Among them, MRI is the most suitable for studying root water uptake, because it is sensitive to the water content (Pohlmeier, Oros-Peusquens et al. 2008). Since it is time-consuming and error-prone for human experts to reconstruct the root structures from the images, root extraction algorithms are developed for automatic extraction. However, the low resolution and signal-to-noise ratio (SNR) of the MRI images often lead to the failure of such algorithms (Schulz, Postma et al. 2012). For the purpose of enhancing the SNR, semantic segmentation can be applied on the MRI images, so that each voxel will be labeled as root or non-root. As for increasing the resolution, image super-resolution methods can be applied, which maps one voxel in the input to multiple voxels in the output. Both semantic segmentation and image super-resolution are well-studied computer vision topics, and the state-of-the-art performance of both is achieved using deep learning methods.

This thesis belongs to the second part of the project: "Advancing structural-functional modeling of root growth and root-soil interactions based on the automatic reconstruction of root systems from MRI" (Schnepf and Behnke 2015). The topic of this part is reducing the noise and improving the resolution in the MRI root images acquired from the first part of the project, to facilitate the extraction of the root structural model in the third part.

For accomplishing both semantic segmentation and super-resolution simultaneously, a previous study used a customized RefineNet (Uzman, Horn et al. 2019), which is a deep learning model originally designed for 2D semantic segmentation (Lin, Liu et al. 2019). Due to the lack of training data and the inaccuracy of the human annotations, the model was trained on a synthetic dataset generated based on the annotations. Each 3D image was converted to a set of 2D image stacks each containing 3 layers, and input to the RefineNet model. Then, the 2D segmentation outputs were assembled into the 3D result. The trained model was able to detect roots from real MRI images with high recall, but also generated a non-negligible amount of false positives. Another previous study to address these tasks used a 3D convolutional neural network (CNN), and the same synthetic dataset for training (Horn 2018). The segmentation results on the real images contained a lower amount of false positives, but the recall of thinner roots was significantly lower than in the results of the RefineNet model, resulting in large gaps in the roots.

Based on the studies above, we hypothesize that more information in the depth dimension can be helpful for reducing false positives, but the synthesized dataset may not have imitated such features well enough. Moreover, due to the memory limitation, the structure of the previous 3D CNN model may be too shallow for the tasks. Therefore, in this thesis, we improve the root synthesis process by incorporating more depth dimension features, such as the aliasing effect of thin roots. Furthermore, real soil images are combined with virtual roots, for integrating more realistic soil noise information into the dataset. Regarding the network structure, a modified version of 3D U-Net is used, which is an image segmentation model that has been shown to achieve relatively good results given a small amount of training data (Çiçek, Abdulkadir et al. 2016). In order to overcome the memory limitation, 3D crops of the original images are used as inputs to the network. We assume that the local information of a crop should be sufficient to decide if a voxel is root or non-root. To speed up the training process, we experiment with importance sampling of the input crops based on their root percentage. Next, we try to make the task easier for the network by letting it ignore the vicinity of the root-soil boundary, because this part of the image is hard to learn and not so important. We do this by first labeling these voxels with the don't-care flag, and then ignore them during loss calculation. Moreover, additional information which may facilitate the network's decision making is added as new input channels, to verify if the information can improve the performance of the network. The types of additional information used are location-dependent information and the image of the same root at a later growth stage. The 2 kinds of location-dependent information are the voxel-wise distance to the pot central axis and the voxel-wise depth. In the end, the segmentation performance on the real MRI images is compared with the RefineNet model, and root structure extraction is performed on the segmentation results of our network.

The thesis is structured as follows: Chapter 2 formally defines the semantic segmentation and super-resolution problems. Chapter 3 introduces some basic concepts about convolutional neural networks. Next, Chapter 4 summarizes the related work, including researches on semantic segmentation, super-resolution, and studies on noise reduction of plant root MRI images. Chapter 5 describes the data generation process, and Chapter 6 describes in detail the method used for image segmentation and super-resolution. In Chapter 7, the results of different experiments are analyzed. Finally, Chapter 8 summarizes the contributions of this thesis and possible future work.

2 Problem definition

The goal of this thesis is to apply semantic segmentation and super-resolution on 3D plant root MRI images, for reducing the noise and increasing the resolution. Because the voxels are to be segmented into 2 classes, root and non-root, it is a binary segmentation problem. Here, the definitions of binary segmentation and super-resolution in our task are given as follows.

Semantic segmentation is defined as mapping each pixel/voxel of an image to a certain class. In our case, a volumetric image V is represented with a 3D matrix of real numbers: $V \in \mathbb{R}^{x \times y \times z}$. The goal is to learn a mapping function f. For each voxel v of image V, f maps it to the probability of v belonging to the positive class (root): $P_{v\_root} = f(v)$, $f(v) \in [0,1]$. Then, a threshold θ is used to decide which class label will be assigned to voxel v: 1 (root) if $P_{v\_root} \geq \theta$, otherwise 0 (non-root).

For super-resolution, a scale factor k needs to be defined first. In our case, the definition of super-resolution can be formulated as learning a function g which maps each voxel $v \in \mathbb{R}$ in image V to a 3D matrix $v' \in \mathbb{R}^{k \times k \times k}$, where v' is a possible super-resolution version of voxel v. As a consequence, the image $V \in \mathbb{R}^{x \times y \times z}$ is mapped to its super-resolution version $V' \in \mathbb{R}^{kx \times ky \times kz}$.

Since we want to achieve both binary segmentation and super-resolution simultaneously, the combined problem is to learn a mapping h that maps each voxel $v \in \mathbb{R}$ of image V to a 3D probability matrix $P_{v'\_root} \in [0,1]^{k \times k \times k}$. Next, a threshold θ is applied on the probability matrix $P_{v'\_root}$ to generate the label matrix $L_{v'}$ in super-resolution: for each element p in $P_{v'\_root}$, if $p \geq \theta$, the corresponding element in $L_{v'}$ is assigned with 1, otherwise 0. In the end, we obtain $L_{V'} \in \{0,1\}^{kx \times ky \times kz}$ for the whole image V as the super-resolution segmentation result.

3 Theoretical background

3.1 Artificial neural networks

The artificial neural network is a powerful method of learning multi-level representations of data, and it has achieved enormous success in various fields including visual object recognition, speech recognition, and machine translation (LeCun, Bengio et al. 2015). While traditional machine learning algorithms needs expert-engineered feature extraction from the raw data, artificial neural networks can work well with the raw data itself.

An example of a very simple neural network is shown in Figure 3.1. The input information is linearly combined by each artificial neuron, and transformed by a non-linear activation function, and then becomes the input to the next layer.This layered structure of connected non-linear modules is

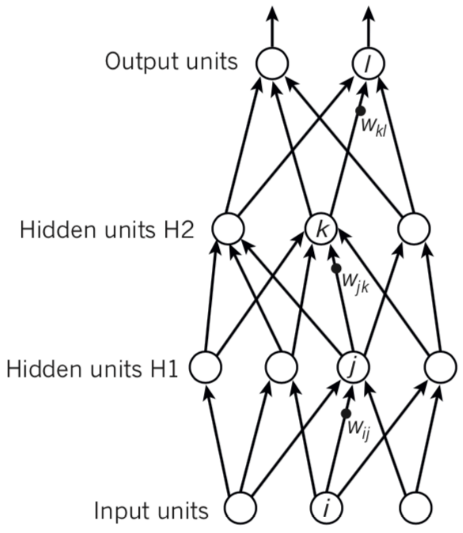


Figure 3.1: An example of a simple multi-layer neural network. (LeCun, Bengio et al. 2015)

responsible for extracting more and more high-level features from the raw data. The higher-level representation layers amplify the aspects of the input that are important for achieving the task. For example, for the image classification problem, the lowest layer usually extracts basic features such as edges, the layer following it extracts simple combinations of edges, and the subsequent high-level layers might represent more complete objects as the combinations of simpler motifs. Usually, methods which use neural networks with multiple layers of neuronal units are also called deep learning methods.

Since deep learning methods need less domain expertise in engineering the features from raw data than classical approaches, with greater computation capacity and larger data, its performance can be easily improved. Moreover, nowadays new network architectures and learning algorithms are constantly being developed, enabling deep learning methods to produce even better results. Thanks to this characteristic, deep learning has made major advances in a broad spectrum of application fields.

3.2 Convolutional neural networks

Convolutional neural networks (CNN) are designed to process array data, which can be of different dimensions, such as 1D for speech (Abdel-Hamid, Mohamed et al. 2012), 2D for images and 3D for volumetric images or videos (LeCun, Bengio et al. 2015). The design of CNN's structure was inspired by the animal's visual neural system, mimicking the concepts of simple cell and complex cell in it (Hubel and Wiesel 1962).

An example of a typical CNN can be seen in Figure 3.2 (Sharma 2018). One basic component of CNN is the convolutional layers. A convolutional layer consists of a number of filters, which are used to convolve with local patches throughout the image (or the feature map from the last layer), producing intermediate feature maps. The function of these feature maps can be seen as capturing the existence of specific features corresponding to each filter.

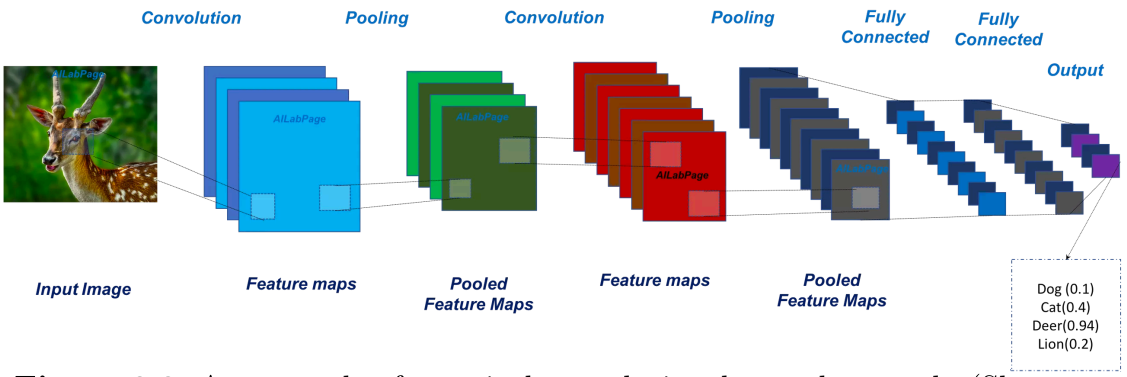


Figure 3.2: An example of a typical convolutional neural network. (Sharma 2018)

There are 2 reasons for using this architecture, firstly it utilizes the fact that local values in array data (such as images) are usually highly correlated, which forms local motifs that can be captured by specific filters. The second reason is that these local motifs can exist anywhere in the image (location invariance), so the same filter is designed to scan and convolve with the whole image to form one feature map.

Another important component of CNN is the pooling layer after the convolutional layer. It usually takes either the maximum or average value of a local patch on the feature map. One reason for doing this is to tolerate minor shifts and distortions of the feature in the image, thus the feature can be detected more reliably. Another reason for using the pooling layer is to increase the receptive field of each output neuron, so that more contextual information can be incorporated. The combination of a convolutional layer, a non-linear activation function, and a pooling layer usually are repeated and concatenated in order to capture different hierarchies of features.

There are a large number of applications of CNN, for example in computer vision problems such as image classification and segmentation. Especially, in the ImageNet competition in 2012, CNN lowered the error rates by half compared to the best competing approaches (Krizhevsky, Sutskever et al. 2012). Since then, it has become a state-of-the-art approach for most of the recognition and detection tasks in the field of computer vision. These big advances were possible thanks to the use of graphics processing units (GPUs) for efficient computation, the non-linear activation function rectified linear unit (ReLU) (Nair and Hinton 2010), a new regularization method called dropout, and data augmentation to generate more training data. The combination of the hardware and software progresses reduced the time required to train CNN from weeks to hours.

Since our problem is to segment the 3D MRI image of a plant root into root and non-root voxels, CNN is the most promising approach to be used in our case.

4 Related work

4.1 Semantic segmentation

Semantic segmentation of images is also called pixel-level classification. As the name implies, the task is to assign a class to each pixel or voxel, and in the end the pixels or voxels of the same object class will be clustered together (Thoma 2016). There are a wide range of applications of semantic segmentation, such as detecting road signs (Maldonado-Bascón, Lafuente-Arroyo et al. 2007), segmenting tumors from brain images (Moon, Bullitt et al. 2002), and tracking medical instruments in operations (Wei, Arbter et al. 1997). The approaches for doing semantic segmentation can be divided into 2 categories: traditional approaches and deep learning approaches. These categories will be briefly introduced below.

Traditional approaches Traditional approaches of semantic segmentation rely heavily on the concept of feature, which is a piece of information that is relevant for solving the problem at hand. Unlike deep learning which can automatically learn the relevant features, these methods require carefully designing the features using domain expertise. There are a large variety of different features that are developed for the semantic segmentation problem, such as Histogram of Oriented Gradients (HOG) (Dalal and Triggs 2005), Bag-of-visual-words (BOV) (Csurka, Dance et al. 2004), and many more.

The traditional approaches of semantic segmentation can be categorized into 2 types depending on the task: unsupervised and supervised ones. In unsupervised approaches, there are no pre-existing labels of the data, which means no ground truth class is assigned to each pixel/voxel. One example is K-means clustering, which aims at partitioning n samples (pixels/voxels) to k clusters. In supervised approaches, there is a ground truth label for each training data point, and the algorithm tries to infer a function which maps data features to the label. Among supervised approaches, undirected probabilistic graphical models such as conditional random field (CRF) achieve the best performance, partly because they also make use of the context information around each pixel/voxel when making predictions (Lafferty, McCallum et al. 2001).

Deep learning approaches Compared to traditional approaches, more recent deep learning methods have significantly enhanced segmentation accuracy. Since CNNs are specifically designed to extract representations from natural images, CNN-based models have become state-of-the-art methods for image segmentation. For example, fully convolutional networks (FCNs) replaces the fully connected part of the network with convolutional layers, allowing the network to produce the output of the same shape as the input (Long, Shelhamer et al. 2015). Another approach used for improving the performance is the dilated convolution, allowing multi-scale contextual information to be aggregated systematically (Yu and Koltun 2015). One advantage of using dilated convolution compared to downsampling-upsampling is that the former does not lead to a loss of resolution. Moreover, there have been some important advances in the backbone architecture, leading to performance breakthroughs. One such example is the use of residual modules in the network (He, Zhang et al. 2016). The function of the residual module is to force the network to learn a residual function, by adding the input of one module to its output. It has been widely tested that the residual model efficiently improves the performance of very deep networks.

4.1.1 Semantic segmentation of 3D images

The above mentioned new methods for getting better segmentation results are mostly applied to 2D segmentation problems. However, there are also needs for 3D image segmentation, such as detecting roots from noisy 3D images. Although the problem of 3D is similar to 2D, there are still some different requirements and constraints. For instance, 3D segmentation generally requires a larger memory for computation. In this section, I will introduce in more detail some deep learning networks specifically designed for segmentation of 3D images.

Currently, there are broadly 2 categories of CNNs used to tackle the 3D image segmentation problems: 2D CNN based and 3D CNN based. The 2D CNN based methods usually split the 3D images into slices, and then generate segmentation result for each slice with 2D CNNs (Havaei, Davy et al. 2017). Although there are plenty of powerful 2D CNNs that achieves good performance on 2D images, they cannot incorporate the depth-dimensional spatial information of 3D images, thus the final results are likely to be suboptimal. Because of this disadvantage, 2.5D methods are developed. They use methods to incorporate some depth-axis information and compress it into 2D slices, and then still use 2D CNNs as the segmentation method (Roth, Lu et al. 2014). Besides that, the more recent use of 3D CNNs overcomes this problem more thoroughly. However, one challenge of using 3D CNN is that the GPU memory requirement for training the neural network is much higher. Thus, it is difficult for the network structure to be as deep as many 2D networks, because of hardware constraints. Up until now, various architectures of 3D CNNs have been proposed, including 3D U-Net, V-Net, and many more. The 3D U-Net is shown to achieve good performance given a relatively small number of training data (Çiçek, Abdulkadir et al. 2016), and many newer 3D segmentation models are developed based on it. In what follows, the structure of the 3D U-Net will be briefly introduced.

3D U-Net The structure of 3D U-Net is illustrated in Figure 4.1 (Çiçek, Abdulkadir et al. 2016). Similar to its 2D counterpart, 3D U-Net has a downsampling encoder module followed by an upsampling decoder. The downsampling is achieved with max-pooling and upsampling with deconvolution. Thus the encoder extracts more and more abstract features, and incorporates broader contextual information. As a result, the final result of the downsampling part is coarse. So in order to produce segmentation results with the same resolution as the network input, the decoder is used to increase the resolution (Long, Shelhamer et al. 2015). Shortcut connections are established between the encoder and decoder, on layers of the same resolution.

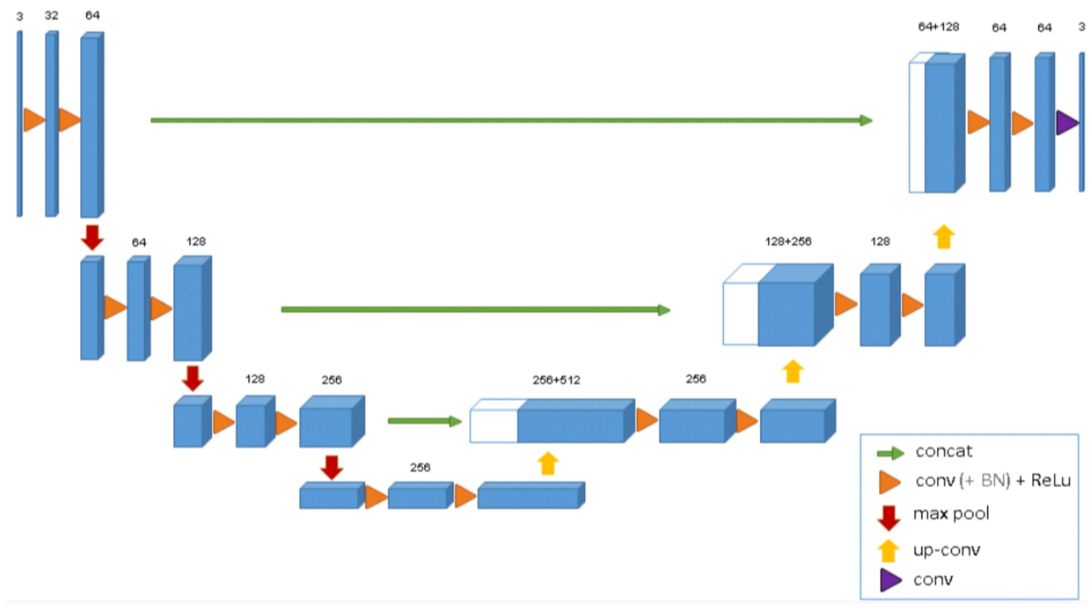


Figure 4.1: The architecture of the 3D U-Net. (Çiçek, Abdulkadir et al. 2016)

These shortcuts are helpful for providing the high-resolution features from the encoder part to the decoder part. Additionally, batch normalization (BN) (Ioffe and Szegedy 2015) is used in the network before each ReLU, for speeding up the learning convergence. BN was designed for mitigating the internal covariate shift which requires a low learning rate for training. With BN, the learning rate can be much higher, and the training becomes less sensitive to the initialization. Moreover, BN can act as a regularization method, helping the model to generalize better. Because the number of annotated training data is small, they are augmented with slight elastic deformations, under the assumption that the images are still biologically plausible. With the combination of the network structure and the data augmentation method, 3D U-Net was able to achieve good results with a relatively small number of data points.

4.2 Image super-resolution

Image super-resolution aims at restoring a high-resolution version of an image given its low-resolution version (Yang, Zhang et al. 2019). It can be categorized into 2 types, single image super-resolution if the input is a single low-resolution image, and multi-image super-resolution if the input consists of multiple low-resolution images of the same scene. The former is more widely used because of its high efficiency. Besides that, because there is only one image for each root, single image super-resolution is used in this thesis. Because images with higher resolution contain more informative details in it, super-resolution is useful in many fields, including medical imaging, satellite imaging and security imaging. However, image super-resolution is an ill-posed problem, because there can be many possible high-resolution images corresponding to the same low-resolution image.

Approaches to tackle the super-resolution task are commonly divided into 3 categories, interpolation-based, reconstruction-based and learning-based. Interpolation-based methods are fast because it's simple and straight-forward, but the resulting accuracy is usually low compared to other methods. One example of it is the cubic interpolation method (Keys 1981). The reconstruction-based methods make use of sophisticated image priors which defines the constraints when reconstructing the high-resolution image. An examples of such constraints is a certain level of smoothness in the output (Dai, Han et al. 2009). The advantage of these methods is that they can generate flexible and sharp details. However, they are usually time-consuming and the quality of the outputs degrades quickly with increasing scale factors. The learning-based methods work by learning the statistical relationship between the low-resolution image and its corresponding high-resolution image from training examples. It can be further classified into traditional machine learning methods and deep learning methods. One example of the traditional machine learning methods is the Markov Random Field (MRF) (Rajan and Chaudhuri 2002). However, similar to the cases of many other computer vision problems, the deep learning methods have shown performance superior to the methods mentioned before. Also, most of the deep learning methods of single image super-resolution are based on CNN.

Super-resolution CNN (SRCNN) is a simple example of a deep learning based approach (Dong, Loy et al. 2014). It first upsamples the input image with bicubic interpolation, and then process the upsampled image with 3 convolutional layers to produce the high-resolution output. Mean squared error (MSE) is used as the loss function. Furthermore, some more advanced methods are developed on top of SRCNN. For example in some works, instead of directly using interpolation, convolutional layers with pooling or dilated convolution are used for downsampling the image, and then deconvolution is used to upsample the image into high-resolution (Dong, Loy et al. 2016). In this way, instead of using a specific interpolation method, the interpolation is learned from the data, resulting in better performance. Another development is to use much deeper network architectures because, in theory, the solution space of the network increases with its depth (Montufar, Pascanu et al. 2014). A different direction for further improvement is to combine domain expertise with deep learning. For example, the combination of sparse coding of images with deep learning has achieved better performance both quantitatively and qualitatively (Wang, Liu et al. 2015). Although many of these optimization-based neural networks can produce results with a high signal-to-noise ratio, they often lack the high-frequency details. In order to deal with this, generative adversarial nets (GANs) have been introduced. One example is the SRGAN (Ledig, Theis et al. 2017), which uses a CNN to generate the super-resolution image, followed by a discriminator network to distinguish between the super-resolution output and the original high-resolution image. The loss of SRGAN includes not only the pixel similarity but also the perceptual similarity, resulting in more realistic outputs. Another advantage of the SRGAN is that it also works successfully with large scale factor, which the other models often fail to achieve.

4.3 Class imbalance

Class imbalance is a problem that can occur in classification, when training samples of certain classes significantly outnumber the other classes. This is problematic because most of the classifier learning algorithms assume a relatively balanced distribution (Sun, Wong et al. 2009). In the case of deep neural networks, when there is class imbalance, it is often observed that the classification error of the majority class decreases rapidly with training, but the error of the minority class increases in the beginning and then decreases very slowly. This is because the gradient is dominated by the majority class (Anand, Mehrotra et al. 1993). This is the case of our root segmentation task, where the non-root voxels significantly outnumber the root voxels. Since image segmentation can also be seen as pixel-level classification, class imbalance is a problem we have to address in this work.

The approaches to deal with class imbalance can be divided into several categories. The first one is data-level approaches, which include different ways of resampling the dataset. Widely used examples include randomly undersampling the majority class (Tahir, Kittler et al. 2009), or randomly oversampling the minority class, or creating new synthetic samples of the minority class (Chawla, Bowyer et al. 2002). Another category is the algorithm-level approaches, which use algorithm-specific modifications to deal with class imbalance. One commonly used example is cost-sensitive learning, which weights minority classes more than majority classes when calculating the loss function (Thai-Nghe, Gantner et al. 2010). The disadvantage of this approach is that one needs to search for the best combination of weights to optimize for a certain evaluation function, which can be computationally expensive. Besides this, loss functions that are designed to be robust against class imbalance can also be used, such as the Intersection over Union (IoU) loss. Since calculating IoU involves thresholding the network's output, IoU is not differentiable. Therefore, directly using IoU as training loss is infeasible. To deal with this, a differentiable approximation of the IoU loss was introduced, which can be used for back-propagation (Rahman and Wang 2016).

4.4 Importance sampling

Currently more and more big datasets enable deep learning methods to achieve state-of-the-art performance in many tasks, but at the same time, training takes significantly longer time with such huge datasets. For example in our case, training the 3D convolutional networks is time-consuming. When training neural networks, it is likely that a large amount of time is spent on the samples that can already be handled well by the network, but the harder samples are not given more focus. By sampling the harder samples more often, importance sampling can help the learning converge faster. The theoretical basis for the speedup by importance sampling is that it reduces the variance of the gradient estimates in stochastic gradient descent, so that the optimum can be reached more efficiently (Alain, Lamb et al. 2015).

There are different ways of doing importance sampling, which can be categorized based on the metrics used for evaluating the 'importance' of each sample. For example, the 'importance' can be the loss or the gradient norm of each sample. One example of using the loss to do importance sampling is ranking the data points by their losses every certain number of training epochs, and then select the data points which have higher losses with higher probability during training (Loshchilov and Hutter 2015). Using gradient norm for importance sampling is a more direct way to reduce the variance of gradient estimates, but it's usually more expensive to compute (Alain, Lamb et al. 2015).

4.5 Root structural model reconstruction

Since the roots are hidden under the soil, some observation methods need to dig them out, wash and scan, or grow the plant in transparent agar (Nagel, Schurr et al. 2006). But all these disrupt the normal growth of the plants. In order to allow observation during growth under more natural conditions, a non-invasive imaging method needs to be used. MRI is one of the methods that allow non-invasive 3D imaging (Brown, Cheng et al. 2014), and the intensity of its signal reflects the water content, which is important for studying root water uptake. Therefore, it is chosen as the imaging method for our project.

One goal of this project is to help automatically reconstruct the root structural model from the 3D MRI images. This is because it is time-consuming to manually inspect each image and extract the relevant biological parameters such as the root length. In one of the recent efforts to automate this process, the algorithm automatically reconstructs the root models from 3D MRI images and provides further root phenotype calculations based on the models (Schulz, Postma et al. 2013). The algorithm consists of 3 steps: firstly, tubular structures are detected in the root image. Then, all root voxels in the image are connected to the root base through the shortest path, using Dijkstra's algorithm. Here the distance measure is the Euclidean distance weighed with the tubularness measure. Finally, the constructed tree is pruned, such as removing root branches that are too short.

An MRI measurement can be done with different spatial resolutions. When done with a low resolution, the measuring time is significantly reduced, but the resolution and the signal-to-noise ratio (SNR) of the results will also decrease (Brown, Cheng et al. 2014). In massive plant root studies that require repeated measurements over an extended period of time, the time cost needs to be considered, leading to a compromise of the resolution and SNR. However, the above-mentioned root extraction algorithm has worse performance when the resolution and SNR decreases. Therefore, enhancing the resolution and SNR of root images without increasing the measurement time is important for guaranteeing the quality of the root extraction. This is the goal of this thesis.

More recently, NMRooting, a modified version of the above-mentioned algorithm, was developed (van Dusschoten, Metzner et al. 2016). It contains some additional features, including dilating the input image to bridge small gaps along root branches.

4.6 Enhancing root image resolution and SNR

As explained before, the resolution and SNR of root images need to be high enough to ensure a decent quality of the root extraction results. For this purpose, 2 research works have been carried out (Uzman, Horn et al. 2019) (Horn 2018). They both use deep learning methods for segmenting root images into root and non-root, thus achieving a higher SNR. Their methods also output the segmentation with a higher resolution than the input.

One work used RefineNet (Lin, Liu et al. 2019) with a pre-trained ResNet module (Figure 4.2) to apply the segmentation and super-resolution on 2D inputs (Uzman, Horn et al. 2019). To transform 3D images to 2D inputs, several consecutive image slices are merged together with principal component analysis (PCA) or averaging, for incorporating more depth dimension information. Because the amount of real data available for their work was very limited, the authors generated an augmented dataset from the annotations of the real data. These data are generated by combining the reconstructed root annotations with synthesized soil images that simulate the real soil features. The results on two real MRI images show that the model is able to detect the root structures quite completely, but produces a non-negligible amount of false positives

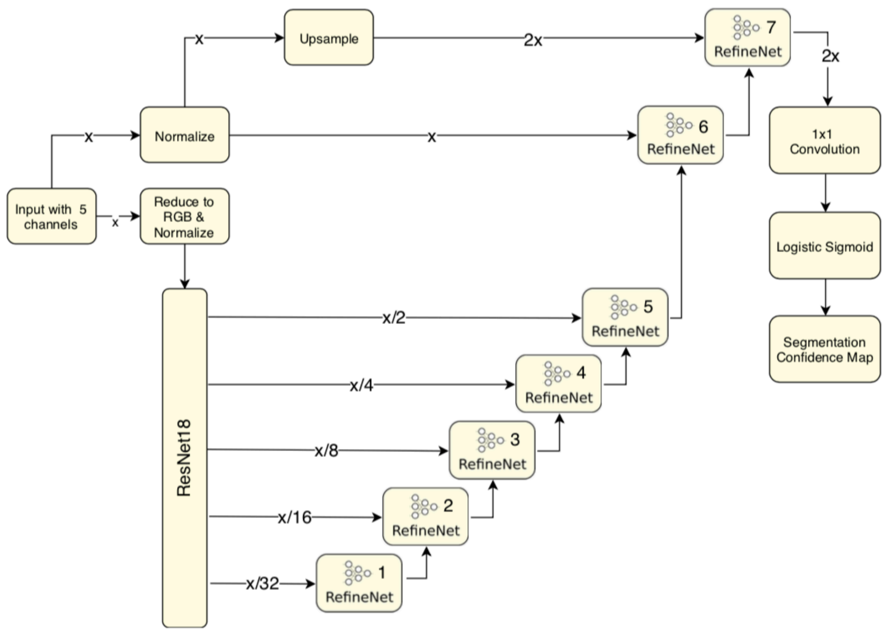


Figure 4.2: The RefineNet architecture for semantic segmentation and super-resolution. (Uzman, Horn et al. 2019)

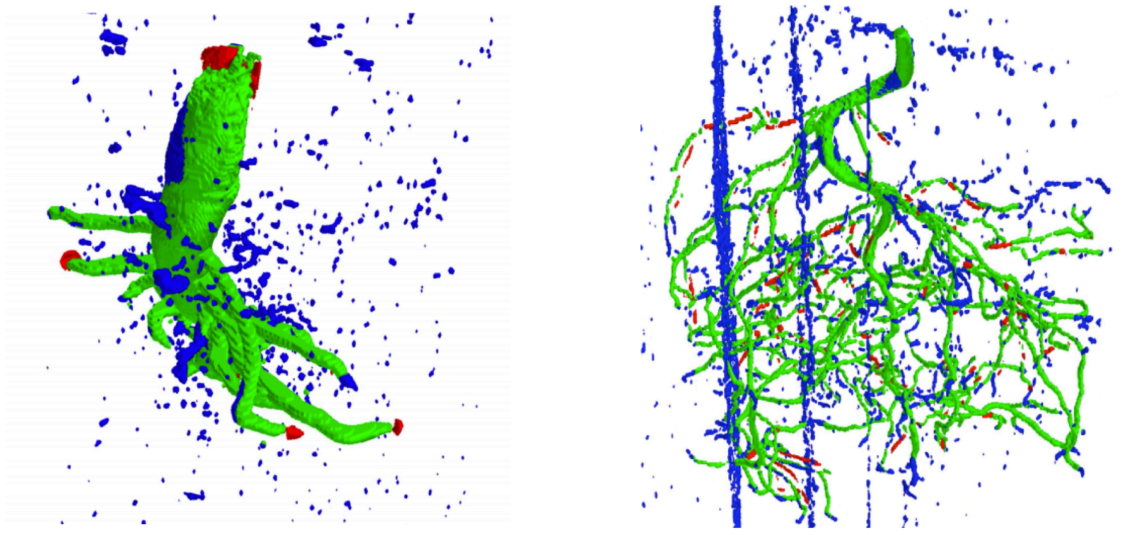


Figure 4.3: The segmentation results of RefineNet on two real MRI images. True positive, false positive, and false negative predictions are marked with green, blue, and red, respectively. (Uzman, Horn et al. 2019)

for the thinner root system (Figure 4.3).

The other work used a 3D CNN (Figure 4.4) and the same augmented dataset (Horn 2018). The model was able to detect the root well for the simpler root image (Figure 4.5, left), but with more false positives in the soil area compared to the result of the RefineNet model (Figure 4.3, left). In the result of the more complicated root image, there is a significant amount of false negatives which makes some root branches disconnected (Figure 4.5, right). Because the GPU memory requirement is high for training a 3D CNN, the

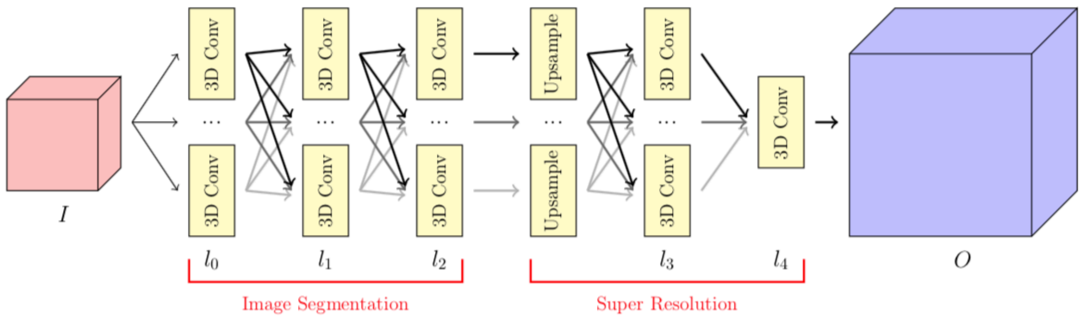


Figure 4.4: The 3D CNN architecture for semantic segmentation and super-resolution. (Horn 2018)

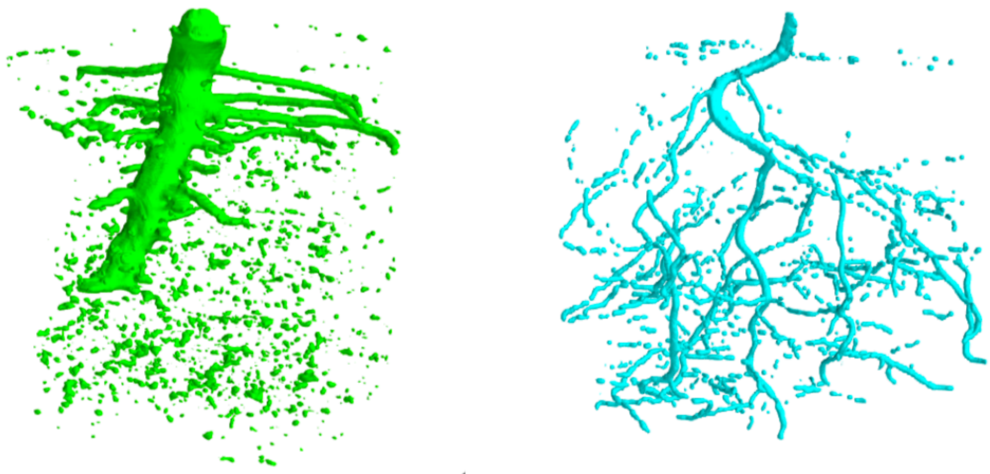


Figure 4.5: The segmentation results of the 3D CNN on the same real MRI images. (Horn 2018)

depth of the network was limited in this work, which may be one of the reasons why the performance is not as good as the RefineNet model. Another possible reason is that the features along the depth-axis in the real images are not well simulated in the augmented dataset, so the network cannot learn such features to help make correct predictions.

5 Data

To experiment on the influence of the diversity of training data, there are 2 data sources used for the experiments, including: 1) the original dataset from the previous work (Uzman, Horn et al. 2019), generated by combining virtual roots and virtual soil data which simulate real soil features; 2) A new dataset of randomly generated virtual roots, combined with real soil data.

In the original dataset, the virtual roots are reconstructed human annotations of 4 real MRI images. These roots are then augmented to increase the diversity, with augmentation operations including radius multiplication, rotation, horizontal axis flipping and swapping. The virtual soil data are generated by simulating the observed features in the real MRI images, for example, the big foggy chunks of noise are simulated with the Perlin noise (Perlin 1985). In the final dataset, there are 384 training samples, 192 validation samples, and 81 visualization samples. The test set consists of 2 usable real MRI images, Lupine Small and Lupine 22.

The generation of the new dataset from random roots and real soil will be described in detail as follows.

5.1 Generation of the new dataset

5.1.1 Caveats of the original dataset

One problem of the generated virtual root in the original dataset is that there is a feature of the real root that was ignored: for some thinner roots which grows in a almost horizontal direction, they appear slightly disconnected (Figure 5.1). This is due to the aliasing effect: each horizontal slice of the real MRI image contains information of a thin horizontal slice of the real 3D space, but the corresponding real space slices of 2 adjacent image slices are not directly adjacent. This leads to disconnections in some roots in the image, because the spatial information is lost between the 2 adjacent image slices.

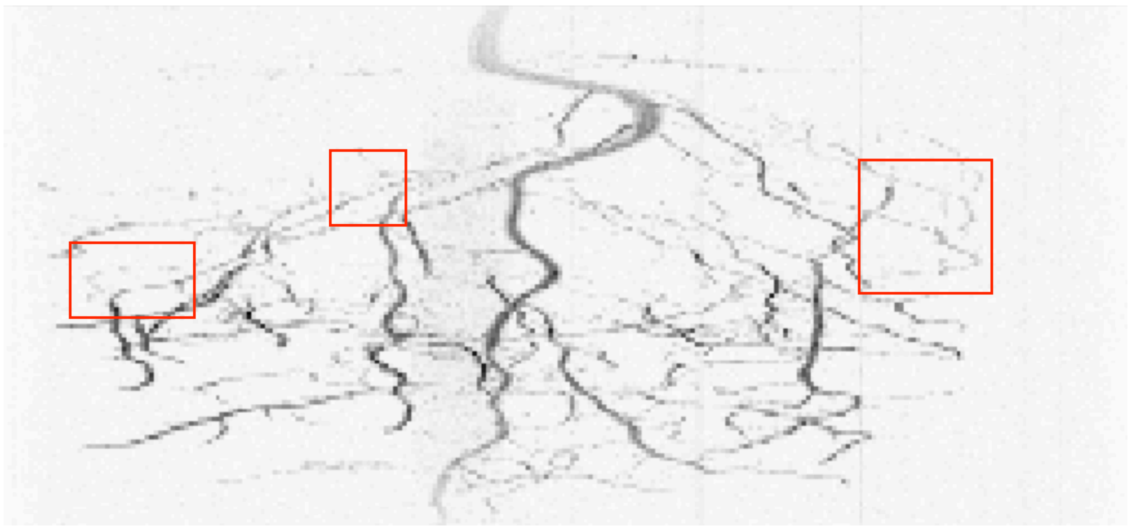


Figure 5.1: The aliasing effect observed in the real MRI image of Lupine 22. Examples of roots with the aliasing effect are marked in red boxes.

This may not influence the 2D convolutional network so much, but might cause problems for 3D network, where the disconnections is obvious. In some experiments we observe that similar disconnections also exist in the test output of the 3D convolutional network, when the test input is one real image with such disconnections (e.g. Lupine 22 in Figure 5.1). And the network used was trained completely on the dataset mentioned in part 1, which fits tubular structure along the root without simulation of such disconnections. Therefore, we speculate that if we add this feature in training data, and use the connected super-resolution ground truth, the network may be able to learn to bridge these disconnections in the input data.

Another problem is that the dataset in part 1 are based on only 4 root annotations, although augmented to increase the diversity, the randomness is still limited. And because the validation and test dataset are also based on the same annotations, there might be the possibility that if there's new test data from a completely different root, the network's performance would degrade. This creates the motivation to generate some random virtual roots that are different from each other while still looking realistic.

5.1.2 Random virtual root generation

In order to deal with the problems of aliasing and potential overfitting, random virtual roots with the aliasing effect to simulate the disconnections are generated. This tool used Gaussian process to virtually imitate the growth of the root. From the point of plant shoot where a initial radius is defined by the user, the root coordinate goes to the next point in the direction of growth with a predefined step size. The initial direction of growth changes after every growth step, according to the sum of a random vector and a gravity vector which always points downwards. Thus the root grows in general downwards with some randomness. The root radius also shrinks by a predefined factor while growing, and then randomly upscaled a little. Once the radius decreases below a predefined small value, or the root grows above or below the allowed range, the growth ends. Along the growth of each branch, there's a predefined probability of branching out a new root at each growth step. The direction of the new branch is controlled by the current growth direction of the parent branch, plus a predefined angle between parent and child branch, and a gravity vector that points downwards. Moreover, if the growing root reaches outside the predefined pot border, the growth step is shortened to keep the root within the pot, and then the growth continues as described above. This makes sure that the root always grows inside the pot. Once the root growth is complete, we obtain a list of anchoring points along the branches, with location coordinates and radii. Based on the list, 3D Gaussian blobs are generated based on the location and radius to voxelize the root structure, which results in the occupancy grid of the root. Afterwards, the ground truth is obtained by binarizing the occupancy grid with a threshold of 30% of the maximum intensity: any voxel with an intensity larger than the threshold gets the root label 1, else non-root label 0.

Besides increasing the diversity of the root structures, we also introduce the aliasing effect in the 1× resolution data which will be used as network inputs. This is achieved by first generating the random root with higher depth axis resolution (6×), and when downsampling to 1× resolution, take 1 slice from every 6 horizontal slices, instead of using the average of the 6 slices. On the other hand, the 2× super-resolution file used as ground truth is obtained by taking the average of every 3 slices, thus generating ground truth without disconnections (Figure 5.2). By providing aliased input data to the network and using super-resolution non-aliased ground truth, we expect the network to learn to bridge these disconnections in the input data.

By adjusting the predefined parameters, different kinds of root structure can be generated, such as ones that look like the real data Lupine Small or Lupine 22. Examples are shown in Figure 5.3. In the end, 10 random Lupine Small-like roots, 10 random Lupine 22-like roots and 10 random roots generated with parameters between the previous 2 root types are created.

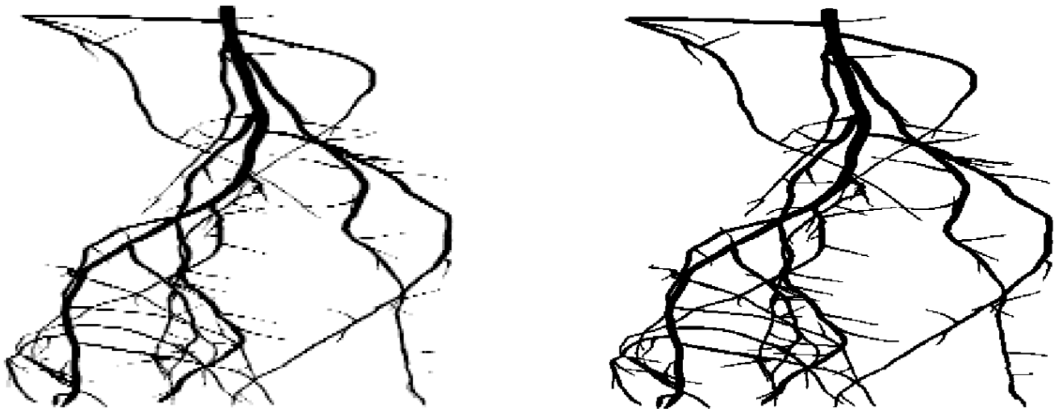


Figure 5.2: The aliasing effect simulated in the random root generation. Left is the aliased 1× resolution root used as the network input, compared with the non-aliased 2× resolution root used as the ground truth (right). Both images are projections on one of the horizontal dimensions.

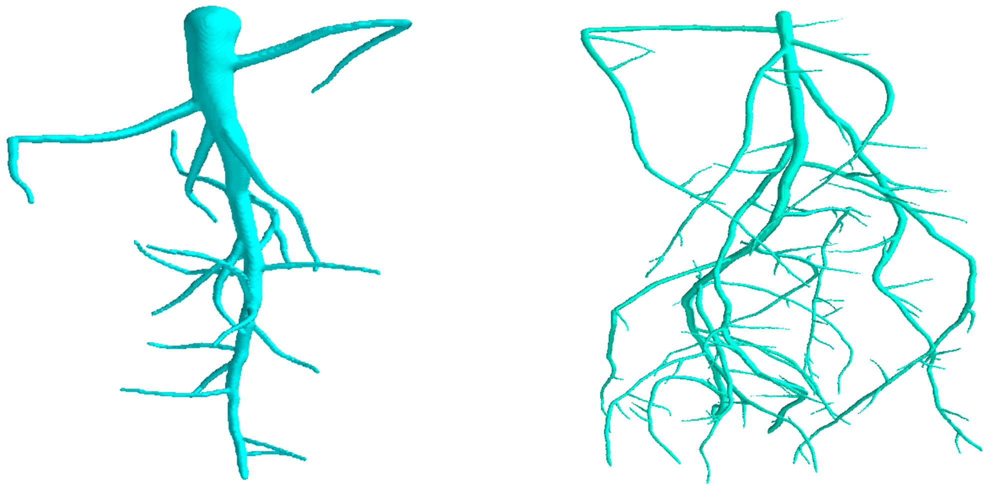


Figure 5.3: Examples of generated root imitating different real roots. The images shown here are super-resolution ground truths, imitating the root structure of Lupine Small (left) and Lupine 22 (right), respectively.

5.1.3 Combining virtual root with real soil

For better imitation of the real data, we use the real soil samples to combine with these virtual roots. There are in total 10 real soil samples, including 8 with the dimension of 256×256×70 and 2 with the dimension of 256×256×121. Since the input to the network is a 3D crop of the whole input data, it needs to be combined from a crop of the virtual root data and a crop of the real soil data. The combination method used is described as follows: Firstly, the root crop is noised with some random Gaussian noise. Then, its brightness is rescaled, with a random factor between [0.8, 1.3]. Next, the root and soil crops are added together to generate the combined crop, where the transparency of the root is control by a factor in the range of [0.7, 1]. Finally, the combined crop is multiplied with a random factor in the range of [0.2, 1.4]. Example horizontal slices of some combined crops are illustrated in Figure 5.4, in which we can observe different contrasts between root and soil, and also different overall intensities.

During training, other augmentations on both the root and the soil crop before combining can be applied on-the-fly, such as flipping one axis and swapping the horizontal axes, further increasing the variety of the training data.

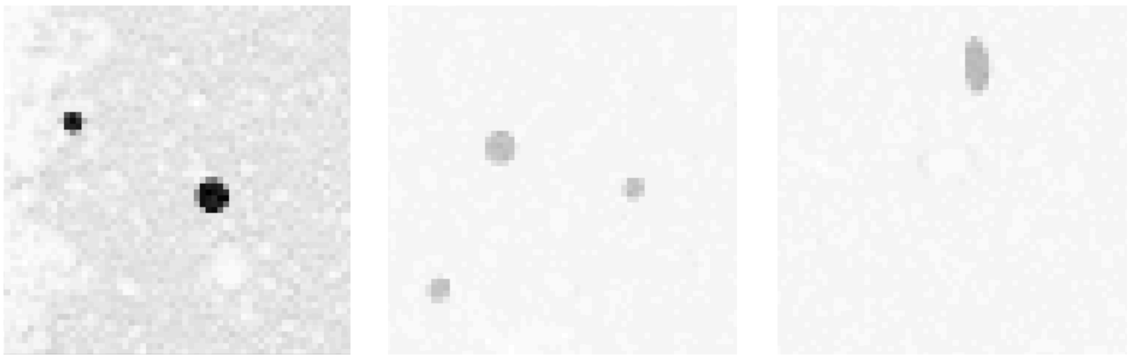


Figure 5.4: Example horizontal slices of combined crops virtual root and real soil. Darker spots in the images are cross sections of the root branches.

5.1.4 Splitting the dataset

Among the 30 random virtual roots, we randomly select 24 roots as training data, and the rest for validation. Because we are only interested in the segmentation result on the real MRI images, we did not use these virtual data as test data. There are 3 possible ways to augment each soil crop, including flipping x axis, flipping y axis and swapping x-y axes. So, for the real soil data, there are 8 (2×2×2) distinct ways of augmentation in total. For each real soil image, we use one specific augmentation of it only during validation (flipping x axis, flipping y axis, and swapping x-y axes simultaneously), and use the other augmentations in training. It is done this way instead of using some real soil images only for validation because the amount of data is highly limited, and it would likely worsen the performance if the variety of training data is sacrificed. Because we randomly take small crops (such as 60×60×60) for training, this dataset can already provide a large diversity of input crops for training.

As will be shown later, the validation loss curves (also the validation F-score curves) during training are roughly equivalent when either using the whole image or crops from those images for validation. Thus, for faster validation, we use the crops in most of the experiments. Because of this, there is no need to generate the whole combined images, but combined crops are enough for the purpose of validation. Furthermore, in order to make use of all the previously generated dataset and thus cover a broader distribution in the datasets, we combine our dataset with the original dataset (Uzman, Horn et al. 2019), in the hope to improve the network's performance. During training, the number of crops from the combined data (new) is made to be roughly the same as the number from the original dataset, to balance the influences of both datasets.

For testing and visualization, we still mainly focus on the real MRI images at hand. Right now, there are more real MRI images provided by our collaborator, but unfortunately due to certain technical difficulty, many annotations of the real images contain large misalignments or errors. Therefore, these images that cannot be used for testing. This is especially the case when the root structure is complex. Thus we try to select some of the real data that has simpler root structure and the misalignments between the annotation and the MRI image are not severe. Finally, except Lupine Small and Lupine 22_August, the following new real data are also included in the test pipeline: I_Sand_3D_DAP5, I_Soil_1W_DAP7 and I_Soil_4D_DAP7 (Figure 5.5).

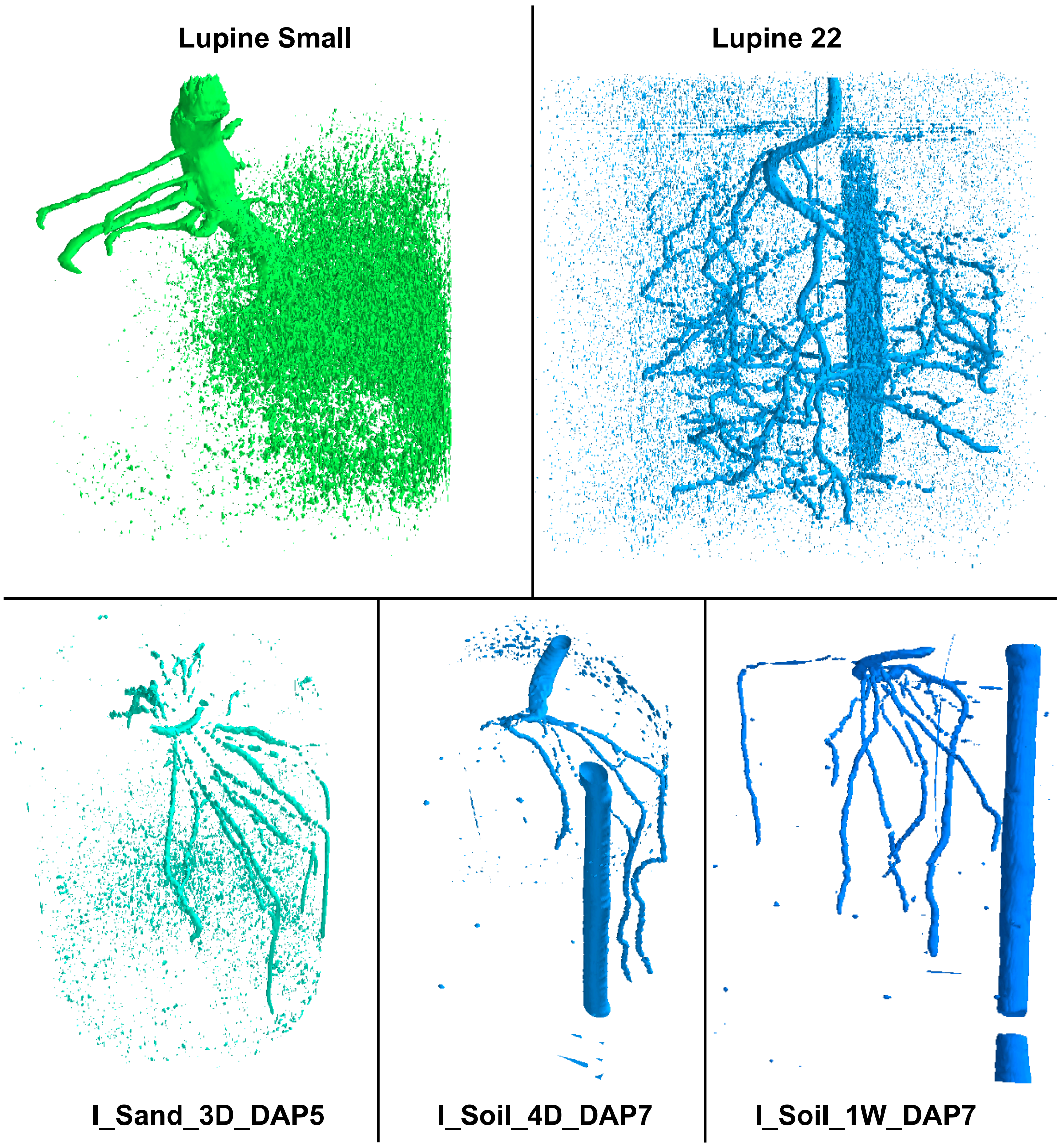


Figure 5.5: The 3D visualizations of the 5 real MRI images in the test set. The cylinders in Lupine 22, I_Soil_4D_DAP7 and I_Soil_1W_DAP7 are test tubes, which are used for calibration when merging image parts.

5.2 Additional input channels

Apart from the noisy root image as network input, adding some additional information as other input channels may help the model extract more relevant features and improve the segmentation performance. We selected the additional information based on if there's a reasoning that it is associated with the distribution of roots. The 2 types of additional information that are tried in the experiments are: noisy root image from a later growth time point, and location dependent information, including the voxel-wise depth and distance to pot central axis.

5.2.1 Simulating roots at different time points

We have a hypothesis that if we can provide the network with not only the root image, but also the same root imaged at a later time point, the network can perform better. This is under the assumption that during growth, the root branches do not change their positions, but only increase the radii and lengths of the root tips. In this case, the earlier root voxels are a subset of the later root voxels, so the network can learn to infer from the later root image to confirm the segmentation of the earlier root image. However, this is difficult to do with the real image. This is because, firstly, there are misalignments between the MRI images and the annotations. Besides, there are also distortions and misalignments between the MRI images of different time points. Therefore, before we can apply this on the real images, it is more reasonable to first test the hypothesis on virtual data with perfect alignment. If providing the later growth stage of the same root helps in this case, it would be worth the effort to try to obtain real images with better alignments, and use the same method to improve the segmentation results.

In order to generate virtual roots of 2 different growth time points, the above method of generating the random virtual roots is used, with some further adaptions. First, the root of the later time point is generated, and the root structure is stored in a tree, where each node represents one branch. The tree hierarchies is the same as the root branch hierarchies, with the children nodes representing the sub branches growing from their parent branch. Each node contains the information of the coordinates and radii of all the anchoring points along the branch, which are needed for voxelization of the branch. Once the whole tree structure is complete, we assume the total growth time is t, and the growth speed of the main branch is v. Then we can calculate the time points at which the secondary branches start growing, based on their branching positions from the main branch. Now with the growth start time point, the total growth time of each secondary branch can be also calculated. Then, the growing speed of each secondary branch can be calculated, because the branch length is known. By iterating this process, the growth start time t_{i_0} and growth speed v_i of any further sub branch i are calculated.

The next step is to generate the same root of an earlier time step. A random ratio r between [0.7, 0.9] is generated, and the earlier time point t_e is assigned with the product of the total growth time t and r. Then we iterate through each node (representing a branch) of the tree, and calculate its length l_e at t_e. After that, we delete the anchor points on the branch which goes beyond the current length l_e. At the same time, the root radius at each anchor point shrinks proportional to the ratio t_e/t. Also, any child node with a growth start time $t_{i\_0} \geq t_e$ is deleted from the tree. Finally, the voxelization is done based on this pruned tree structure, obtaining a shorter and thinner root as outcome (Figure 5.6).

Once we have obtained the roots of different growth time, they are combined with random crops of real soil image as described before. For both time points, the random crops are taken from the same real soil image, to ensure that the soil noise distribution between the time points are similar. This is done because we assume that the soil content would not change too drastically during root growth.

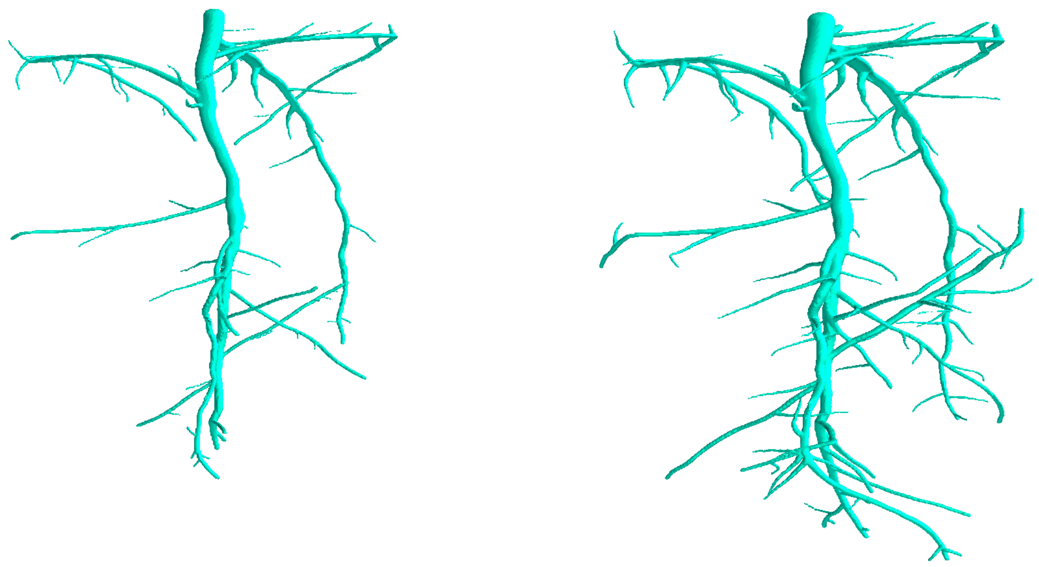


Figure 5.6: The generated root at different time points. Left and right are the same root, at an earlier and later time point, respectively.

5.2.2 Location dependent information

Another assumption is that the location of the root in the pot may follow certain patterns. For example, we observe that many root branches tend to grow horizontally in the beginning, and once they reach the pot border, they grow along the border. Therefore, the location information as additional information channel may provide useful hints of the probability of root existence, thus may facilitate the network's learning process. 2 types of location dependent information are tested in the experiments, the voxel-wise depth and the distance to the pot central axis.

The voxel-wise depth information is used, because we observed that in some real data, the noisiness of the soil seem to change along the depth axis. This changing noisiness may be the effect of gravity on the soil water content. Furthermore, root structures may look different depending on the depth. The depth information is calculated for each voxel input the input image crop. Therefore, it is a 3D array of the same shape as the input crop, with the same value for each horizontal slice.

The voxel-wise distance to the pot central axis is used because of the above mentioned observation that many roots seem to follow the vicinity of the pot border when growing, far from the pot central axis. This phenotype is also simulated to some extent in the random root generation (in Section 5.2.2). Because the location of the pot central axis is known for the generated roots, for each voxel of the input image crop, the Euclidean distance to the central axis is calculated. The result is also a 3D array with the same shape as the input crop. The reason to use the distance to the central axis instead of the distance to the pot border is that, although the latter is easy to calculate for the generated virtual roots, it is hard to calculate for the real root images, because of the distortions of the real images. In the real images, the pots are not in a perfect cylinder shape but rather stacked barrels with changing radii. Moreover, the pot shape may appear elliptic instead of circular in one horizontal slice of the real image (Figure 5.7).

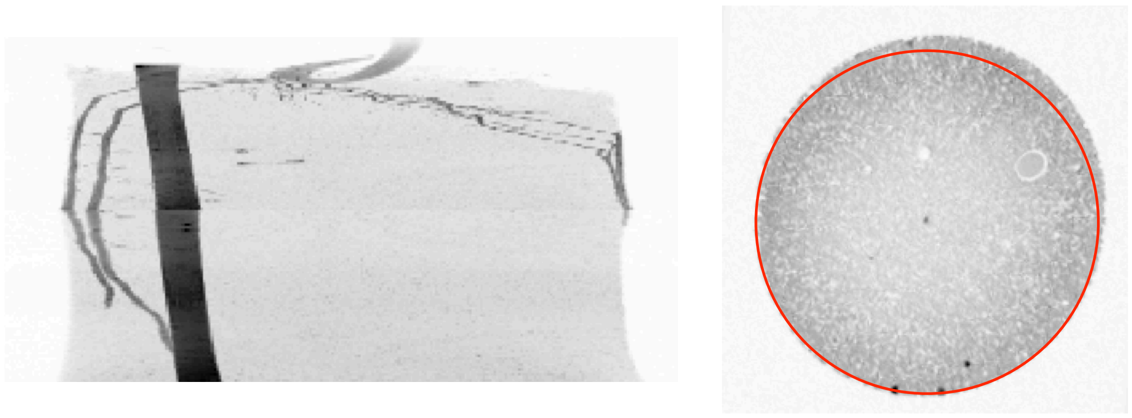


Figure 5.7: The pot shapes in real images. The left is a 3D image projected on a horizontal dimension, showing changing radius along the depth axis. The right is one horizontal slice of one real image, showing a elliptic pot shape (compared to the circle in red).

6 Segmentation method

6.1 Network: 3D U-Net

CNNs has been shown to be the most successful choice for solving computer vision problems such as semantic segmentation and super-resolution, since they are originally designed to extract hierarchical visual features. Although the 2.5D CNN approach of a previous study (Uzman, Horn et al. 2019) has achieved decent results, we hypothesize that the depth dimensional information which is difficult to be fully utilized in 2.5D CNNs may be important for improving the segmentation result. Therefore, 3D CNN is experimented in this work. Furthermore, because now we are using small crops of the whole image as network input, the memory consumption is much lower, allowing the use of deeper network structures than another study that also uses a 3D CNN (Horn 2018). As described in the related work section, 3D U-Net is the base network structure which inspired many further network variations successfully used in 3D image segmentation. It's relatively simple, but still powerful enough to achieve decent performance when the number of training data points is limited. Thus we choose it as the network for our task.

6.1.1 Network structure

The original network structure contains a downsampling encoder and an upsampling decoder part, as described in the related work section (Figure 4.1). The same basic structure is used in our network, with some minor adaptions (Figure 6.1). The input to the network is a 3D crop of a noisy root image. In the encoder part of network, the input goes through 3 convolutional modules, with maxpooling layer (red arrow) between each pair of adjacent convolutional modules. Each convolutional module consists of 2 convolutional layers with increasing channels numbers. After the final convolutional module in the encoder part, the tensor is upsampled using 2× transposed convolution to double the size, keeping the channel number constant. Then the upsampled tensor is concatenated in the channel dimension with the output of previous

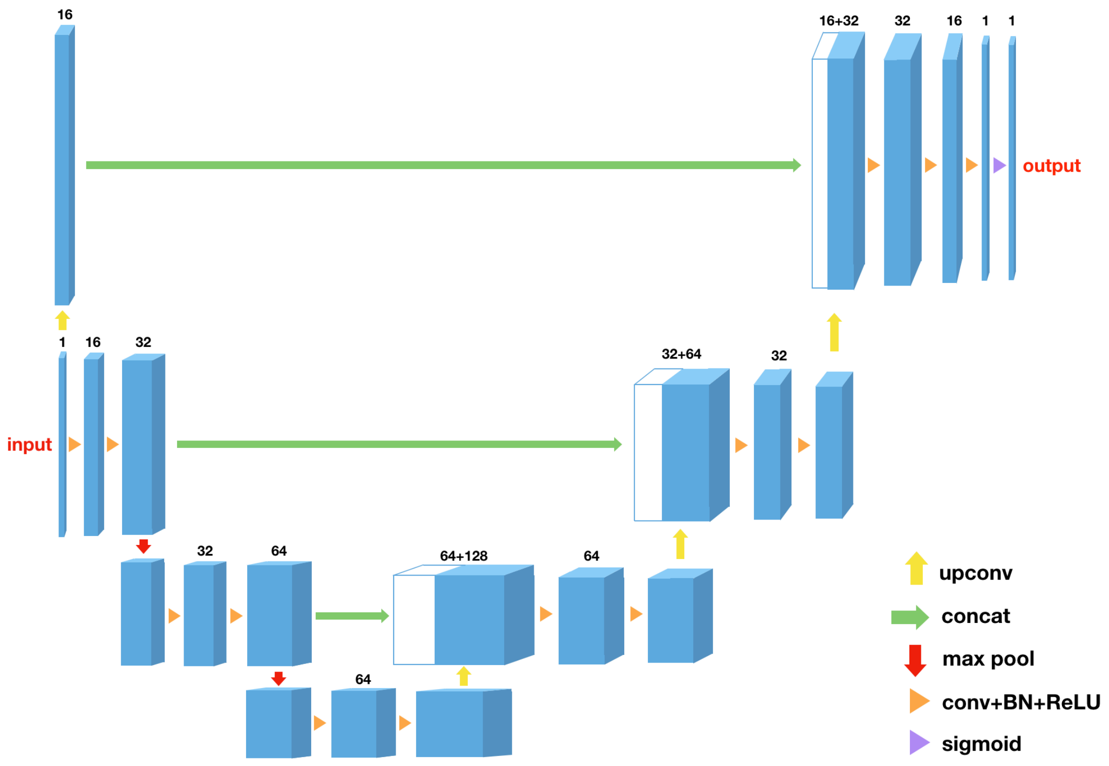


Figure 6.1: The modified 3D U-Net structure used in this work. The numbers above the tensors represent the number of channels. The final output doubles the resolution of the input.

convolutional module which has the same image resolution. The concatenated tensor then goes through 3 convolutional modules with decreasing channel numbers, each containing 2 convolutional layers. Adjacent convolutional modules in the decoder part of network are connected with 2× transposed convolution layers for further upsampling to reach the final 2× resolution of the input. In the end of the network, the number of channels is reduced to 1, and followed by a sigmoid function which outputs probability values between (0, 1).

6.1.2 Valid convolution

For the convolutional layers in this network we tried both padding and not padding the input. No padding during convolution is also called valid convolution, in which case no additional information which does not belong to the original input image will be introduced to the network. We expect that by using valid convolution, the network would not be confused by unrealistic information, thus producing better results. However, one outcome of valid convolution is that the output of the network is smaller than 2× the input size, because without padding, the tensor size decreases a little after each convolutional layer. So the more no-padding convolutional layers there are, the greater the decrease of the output tensor size. In our case, when the size of the input crop is 60×60×60, the output crop size will be 34×34×34, much smaller than 120×120×120 when using padding. During visualization and testing, this leads to the need of cropping the whole image into overlapping crops, so that the output crops can be assembled into the whole 2x resolution image without gaps.

6.2 Training pipeline

6.2.1 Data loading: random crop sampling

Unlike the previous work (Horn 2018) which also used 3D CNN for segmentation, the network input used in this work is 3D crops of the whole image, instead of the whole image or big chunks of the whole image. In this way, we can use deeper neural networks under the constraints of limited GPU memory. Based on observations in the early data exploration, we think the local information in the crop of a reasonable size (such as 60×60×60) already contains a sufficient amount of information for deciding if a voxel is root or non-root. The context information is important for making correct decisions, but the context that's too far away from the voxels of interest does not play a part in whether these voxels are root. Moreover, when using a big input, due to memory constraints, the network structure has to be shallow. For a shallow network, one voxel in the output corresponds to a relatively small receptive field in the input, thus the broader context cannot be utilized by it. Therefore, during training, we randomly sample crops from the training images. Two different methods are used for random sampling: importance sampling and uniform sampling.

6.2.1.1 Uniform sampling

Uniform sampling is a simple sampling method, which is done by randomly drawing the location of a certain vertex of the crop, so that each crop has the same probability of being chosen. This is ensured by sampling the coordinate of each dimension of the vertex from a uniform distribution. With uniform sampling, any part of the training images has the same probability of becoming the input of the network.

6.2.1.2 Importance sampling by root percentage

On the other hand, importance sampling samples data with varying probabilities depending on their importances. Here in this work, the basis for importance sampling is the percentage of root voxels within each crop. For a given whole image, crops which contain more root voxels are sampled with higher probability, so that the visibilities of them to the network are increased. Our reasoning for doing this is that compared to the large varieties of the soil noise signals, the root has a more clear structural pattern to be learned by the network. When we use the crops with more root voxels more often, the network may learn to achieve a higher ability to detect the distinctive structural features of the roots. This may lead to a better segmentation result and less sensitivity to unknown soil noise varieties. Another potential benefit of using importance sampling is that, it may speed up the convergence of the training process, by making the network focus more on the more important training samples, thus wasting less time on easy or unimportant samples.

The critical part of implementing importance sampling is to obtain the crop importance (in this case root percentage) distribution for each image, from which the sample crop will be drawn. This is done by calculating the root percentage of every possible crop of one image and generate a multinomial distribution in which the probability assigned to each crop is proportional to its root percentage. And then randomly sample one crop from this distribution.

When generating the probabilities of the multinomial distribution, a small offset value is added to the calculated root percentages. It's done this way because otherwise, the probability corresponding to image crops without root voxels will be 0, then the network never has the chance to learn from them. We don't want the sampling process to completely ignore the crops without root, because only learning from positive samples may lead to a high false positive rate in the end. And this offset is one hyperparameter that can be adjusted. The higher the offset, the higher the chance of sampling pure soil crops without root, as shown in Figure 6.2. In this figure, for each voxel, the number of rounds it gets sampled is color-coded, the warmer color meaning a higher number. Here, the dark blue areas represent the voxels that are sampled at least 4 times. We can observe that for an offset of 1/255, the closer one voxel is to the root ground truth (darker shape at the center of each image), the higher number of rounds it gets sampled, and only a small proportion of the peripheral area which does not overlap with the root gets sampled 4 or more times. However, when using a higher offset 10/255, most of the peripheral area gets sampled at least 4 times, and the central area which overlaps with the root is being sampled less frequently than before. This shows that by adjusting the value of the offset, we can balance the sampling frequency of the root containing crops and pure soil crops.

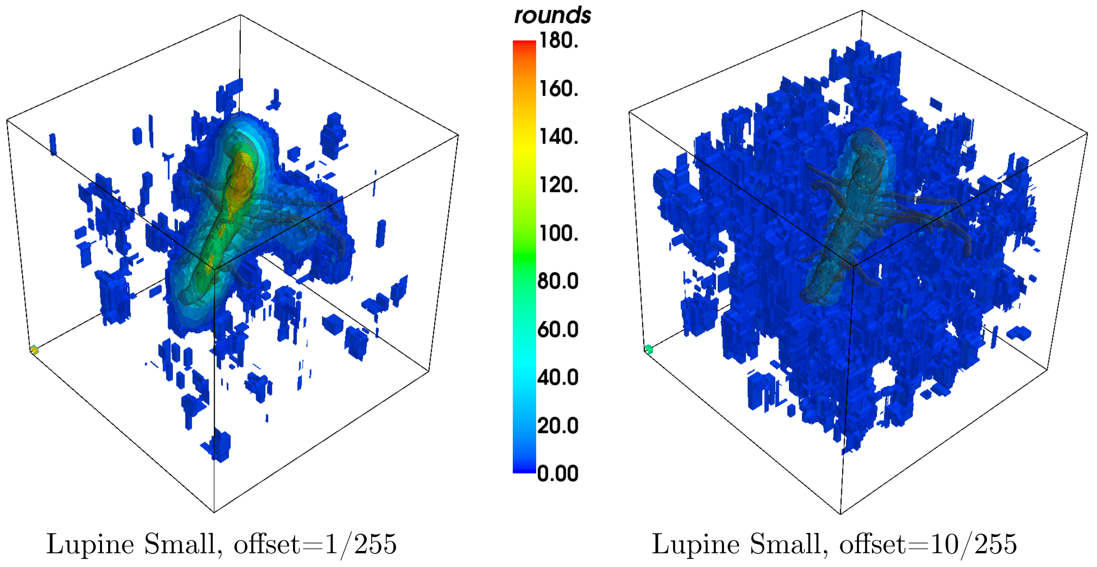


Figure 6.2: The distributions of sampled crops when using different offset values in importance sampling. There are in total 2000 sampling rounds in a Lupine Small image, with a crop size of 60×60×60. Colors denotes the number of rounds each voxel gets sampled. Darker shape in the center is the ground truth root of Lupine Small.

6.2.2 Loss function

6.2.2.1 Weighted binary cross entropy loss

For our segmentation and super-resolution task, binary cross entropy (BCE) loss is used as the loss function. It is the simplified binary version of the negative log-likelihood (NLL) loss, which is commonly used as the loss function for training multi-class classification models. For a single input data i, the formula of BCE loss is given as follows:

$$L_i = -(y_i \log_2(p_i) + (1 - y_i)\log_2(1 - p_i)) \tag{6.1}$$

Here, y_i is the ground truth class (1 if root else 0), p_i is the estimated probability that the input data i belongs to the positive class 1. If the true label is 1, when p_i approaches 1, the loss approaches 0. When p_i approaches 0, which means the model makes a wrong prediction, the loss increases and the rate of increase gets much higher as p_i getting closer to 0 because of its logarithmic property (Figure 6.3). This logarithmic property overcomes the saturation problem of the Sigmoid function used at the end of the network. The Sigmoid function has a very small gradient when the input to it is not in the vicinity of 0, making the network optimization too slow. By overcoming this problem, the BCE loss facilitates the training process.

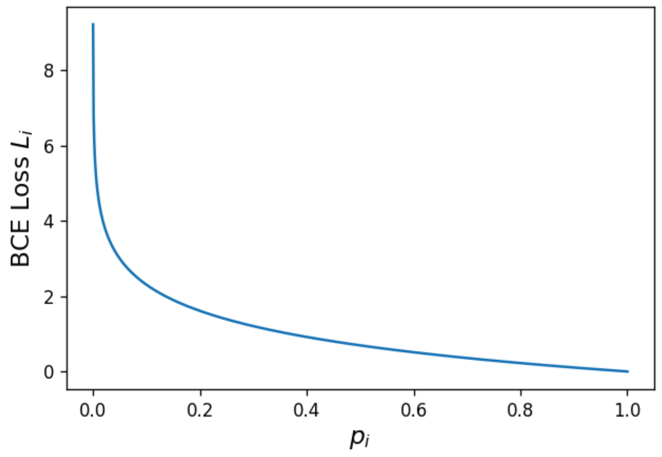


Figure 6.3: The BCE loss with respect to the probability estimation. p_i represents the estimated probability that sample i belongs to the class 1, when the true label is 1.

Formula 6.1 is for a single input data (i.e. one voxel), and for a batch of input images, the BCE loss is calculated as the average loss among all the voxels in it:

$$L = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log_2(p_i) + (1 - y_i)\log_2(1 - p_i)) \tag{6.2}$$

in which N represents the total number of voxels in the mini-batch. However, since in our dataset, the number of non-root voxels heavily outnumbers the root voxels, if we directly use the naive BCE loss as in formula 6.2, the contribution of the non-root voxels will be significantly larger than the root voxels, which may force the network to focus more on improving the prediction accuracy of the non-root part. This imbalance problem might lead to results with a high false negative rate, which tends to have root voxels predicted as non-root. In order to deal with this problem, we introduce a weighting factor for the root voxels in the BCE loss:

$$L = -\frac{\sum_{i,j,k}(Y^{i,j,k} \cdot \log_2 P^{i,j,k} \cdot rw + (1 - Y^{i,j,k}) \cdot \log_2(1 - P^{i,j,k}))}{\sum_{i,j,k}(Y^{i,j,k} \cdot rw + (1 - Y^{i,j,k}))} \tag{6.3}$$

in which rw is the root weight, $Y^{i,j,k}$ is the ground truth label of voxel with coordinates (i, j, k), and $P^{i,j,k}$ is the probability estimated by the network that it belongs to the positive class.

One special case when calculating the weighted BCE loss is when using importance sampling, in which the sampled crops would contain significantly more root voxels than the expectation in the whole data. In order not to bias the model towards predicting too many false positive roots, the loss of each crop in one mini-batch is divided by its relative importance before calculating the average loss of the whole batch, as shown below:

$$I_c = \frac{P_{c\_imp}}{P_{c\_uni}} \tag{6.4}$$

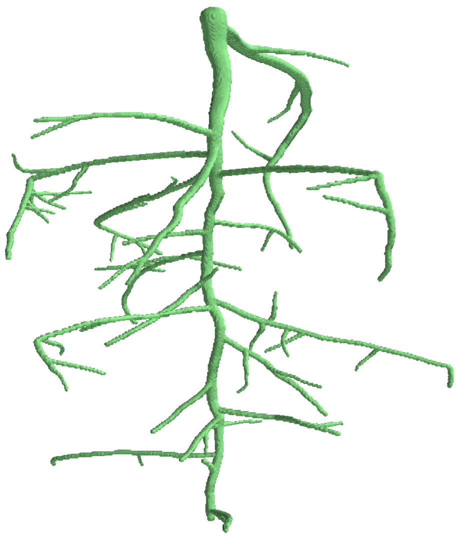
$$L_{batch} = \frac{1}{N}\sum_{c=1}^{N}\frac{L_c}{I_c} \tag{6.5}$$

where I_c is the relative importance of crop c, P_{c_imp} and P_{c_uni} are the probabilities of sampling the crop by importance sampling or uniform sampling, respectively. L_{batch} is the average loss of the whole mini-batch consisting of N number of crops, and L_c is the BCE loss of crop c. For crops containing higher percentage of root, the loss of it will be divided by a larger relative importance, thus contributing less to the computation of gradient during backpropagation.

6.2.2.2 Don't-care flag

Due to the noisiness of the input image, it is difficult for the network to learn to accurately distinguish the borders between the root and soil, so it takes a long training time to refine them. Moreover, there is a problem with the virtual data we use for training: currently we use a threshold of 0.3 to decide which voxels in the occupancy grid should be labeled as root (see 5.2.2), but this threshold choice is arbitrary. This could be confusing information to the network. Even if we label all voxels with positive intensity as root, the root voxels at the root-soil border will have low intensity, so adding noise to it will make these voxels too hard to be recognized correctly as root by the network.

At the same time, our collaborating plant scientists do not require the root surface to be extracted precisely but are more interested in the completeness of the 3D root structure extraction. Therefore, we can adjust the network to ignore the unnecessary confusing details and focus on more important aspects. For this purpose, we introduce the so-called don't-care flag to mask the vicinity of the border between root and soil. And when calculating the training loss, voxels within this mask are ignored. By doing this, the optimization process ignores this difficult part and updates the network based on the more reliably labeled parts of the input image. This mask is generated by labeling voxels close to the root-soil border with 1, on both the root side and the soil side. Any other voxels are labeled with 0. Here, the definition of the root-soil border is the boundary between zero and non-zero in the occupancy grid. Figure 6.4 shows one example of the don't-care mask.



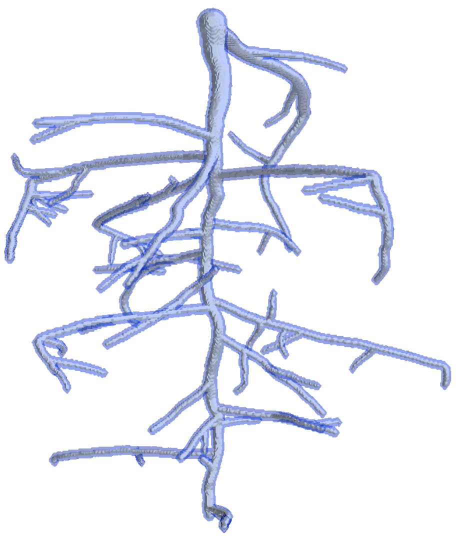


Figure 6.4: Occupancy grids of a random root (left) and the don't-care mask of it (right). The don't-care mask is marked with blue shade, illustrating the border area between root and non-root.

6.2.3 Other training settings

The implementation of this work is done with PyTorch (Paszke, Gross et al. 2017), because the dynamic graph it uses provides transparency on runtime, which makes it convenient for keeping track of what is going on during training and easy to debug. Moreover, it is easy to distribute computation work among multiple GPU cores with PyTorch.

Optimizer Adam (Kingma and Ba 2014) is chosen as the optimization algorithm for training the network, because of its stability of performance and fast convergence speed compared to stochastic gradient descent. 3 different initial learning rates are experimented, 1e-3, 1e-4 and 1e-5. Default values are used for the beta1 and beta2 coefficients, which are 0.9 and 0.999, respectively.

Training epoch In each training epoch, the number of data points (image crops) used is defined as a hyper-parameter before training starts. By default, the number 100000 is used. Each training crop is obtained by randomly sampling one data from the training set, and from the data randomly sampling one crop using either uniform sampling or importance sampling as described before (see 6.2.1). The random crops used in different epochs are the same, but with shuffled order.

6.3 Performance evaluation

6.3.1 Validation

Validation on random crops After a certain number of training batches, we validate the network's performance on some validation data, in order to check if the model is overfitting. When for a certain amount of time the validation loss stops decreasing or starts increasing while the training loss continues to decrease, the training process is considered to have converged and will be manually stopped. Usually, the validation is done on the whole images of the validation set, which takes a considerable amount of computation time. In order to speed up the process, we use one random crop from each validation data for validating the network instead. As long as the crop is uniformly randomly sampled from the data, the expectation of the validation loss of it should be the same as that of the whole image. It can be observed that despite small fluctuations, the overall validation loss curve is roughly the same. Therefore, in the following experiments, we use the random crops for validation.

Validation metrics Besides the root weighted BCE loss which is computed the same way as training data, the validation F1 score is also calculated to evaluate the changes of network performance on the validation data, for that it is relatively robust against class imbalance. The formulas are shown below:

$$precision = \frac{TP}{TP + FP} \tag{6.7}$$

$$recall = \frac{TP}{TP + FN} \tag{6.8}$$

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{6.9}$$

Here TP/FP/FN represent the number of true positives, false positives, false negatives, respectively. Since the output of the network is the estimation of the probability of each voxel being root, which are continuous values between 0 and 1, a threshold of 0.5 is used to binarize the output into positive and negative predictions. When calculating the validation metrics of whole images, the F1 score is calculated for each whole image, and then averaged to get the F1 score for the validation dataset:

$$F_{1\_avg} = \frac{1}{|V|} \sum_{v \in V} F_{1\_v} \tag{6.10}$$

Here, V is the validation dataset, v is one image in it, and |V| is the total number of images in V. But when validating on random image crops, the TP/FP/FN are accumulated for all the image crops, and the F1 score for the validation dataset is directly calculated based on them. This is done because, for a large number of random crops, there will be no roots in them, then the number of TP is 0. In this case, both the numerator and denominator of the F1 score are 0, making the calculation not valid. However, if TP is accumulated for all random crops in the validation dataset, when the validation set size is big enough, TP will almost certainly be positive in the end.

6.3.2 Testing

Unlike validation, testing is done on the whole images of the real test dataset. Currently, there are 5 real MRI images used for testing, including Lupine_Small, Lupine_22_August, I_Sand_3D_DAP5, I_Soil_1W_DAP7, and I_Soil_4D_DAP7. F1 score is also used in testing performance evaluation, but because of the misalignments between the human annotations and the real images, the TP/FP/FN numbers cannot be directly calculated. To deal with this problem, distance tolerant F-score is used as described in a previous study (Uzman, Horn et al. 2019). The basic idea of it is to tolerate minor misalignments by ignoring the false positives which lie close to the ground truth root, and the false negatives which lie close to the predicted root. This is done by making the ground truth bigger when calculating precision, and making the root prediction bigger when calculating recall (Figure 6.5). As shown in the figure, both the distance tolerant recall (the percentage of the brown area in the black box on the left) and precision (the percentage of the brown area in the black box on the right) increased compared to original results without dilation. The distance tolerance can be adjusted by changing the

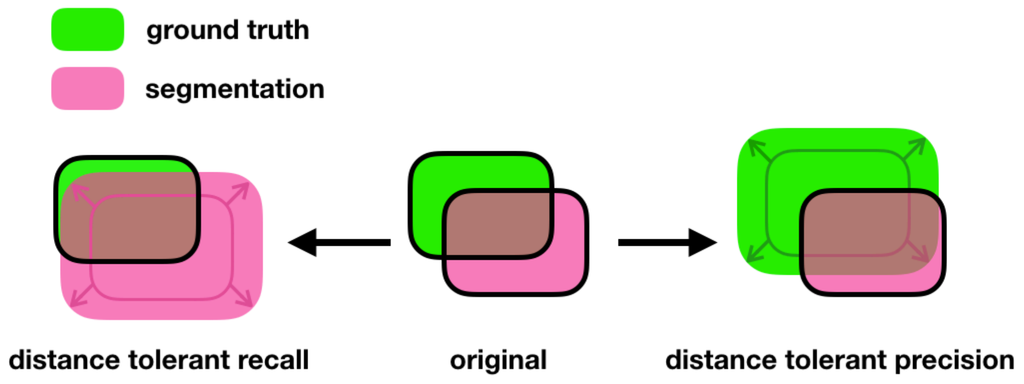


Figure 6.5: Illustration of the calculations of distance tolerant recall and precision. The left shows the dilation of the segmented root to calculate the distance tolerant recall, and the right shows the dilation of the ground truth root to calculate the distance tolerant precision. Green represents the ground truth root, pink represents the segmented root, and brown denotes the overlapping between green and pink, which is the numerator in both recall and precision calculations. The black border delineates the denominator in either recall (left) or precision (right) calculation.

dilation distance. Since there is no best distance tolerance for calculating the test F score, multiple distance tolerance values will be used, and the model's performance on the test data will be evaluated by analyzing the plot of distance tolerance F scores against increasing distance tolerances.

In testing, the test tube (a tubular object inserted in the soil, used for calibration) part is ignored by masking the tube region with a zero mask, in both the ground truth and the segmentation output. This is done because the trained models tend to predict the test tubes in real images as roots. The reason for this is that the test tubes in the synthetic training data are always perfect cylinders, which is not a realistic simulation of the real images. The real test tubes are often twisted and with uneven thickness. Therefore, when trained with this dataset, the network learns to segment a test tube as non-root only when it's a perfect cylinder. On the other hand, it is easy to manually remove the test tube from the real images because it has a fixed location and radius, so the correct prediction of the test tube as non-root is not necessary. Therefore, we just ignore the test tube region during testing.

7 Results

In most of the models, we use a training mini-batch of 25, and 100000 random image crops (60×60×60) from the training set in each training epoch. Thus, there are 4000 mini-batches in each epoch.

7.1 Effects of using the combined dataset

As described in section 5.1, to the original dataset of synthetic noisy root images, we added randomly generated roots combined with real soil images. To demonstrate the effect of using the combined dataset, here we compare two 3D U-Net models, one trained with the original dataset (denoted as Model O), the other trained with the combined dataset (denoted as Model C). All other hyperparameters of the 2 models are the same, which are already fine-tuned to ensure relatively good performance. In the combined dataset, the number of training input crops from each dataset is the same. Both models can reach high validation F1 scores after training convergence, which is 0.967 for Model O and 0.964 for Model C (table 7.1). However, the validation set used to validate the 2 models are not entirely the same, because Model C was also validated on some combined data. Therefore, we compare their performance based on the segmentation results of the real MRI images, which are the same for both models.

The visualization of the real Lupine Small data shows that the result from Model O has significantly more false positives in the soil area (Figure 7.1, left). This may suggest that the soil noise simulation in the original dataset does not mimic the real noise well enough in the depth dimension. In comparison, Model C is able to produce segmentations with fewer false positives (Figure 7.1, right). The quantitative comparison with the distance tolerant F1 scores also shows that Model C (orange curve in Figure 7.1) has a better segmentation on Lupine Small. This improvement of Model C is probably because it's also trained with real noise data, so it learns from more diverse and realistic soil noise. When the false positive rate is too high, non-existing branches may be extracted in the root extraction algorithm. Such a risk can be reduced when the combined dataset is used for training.

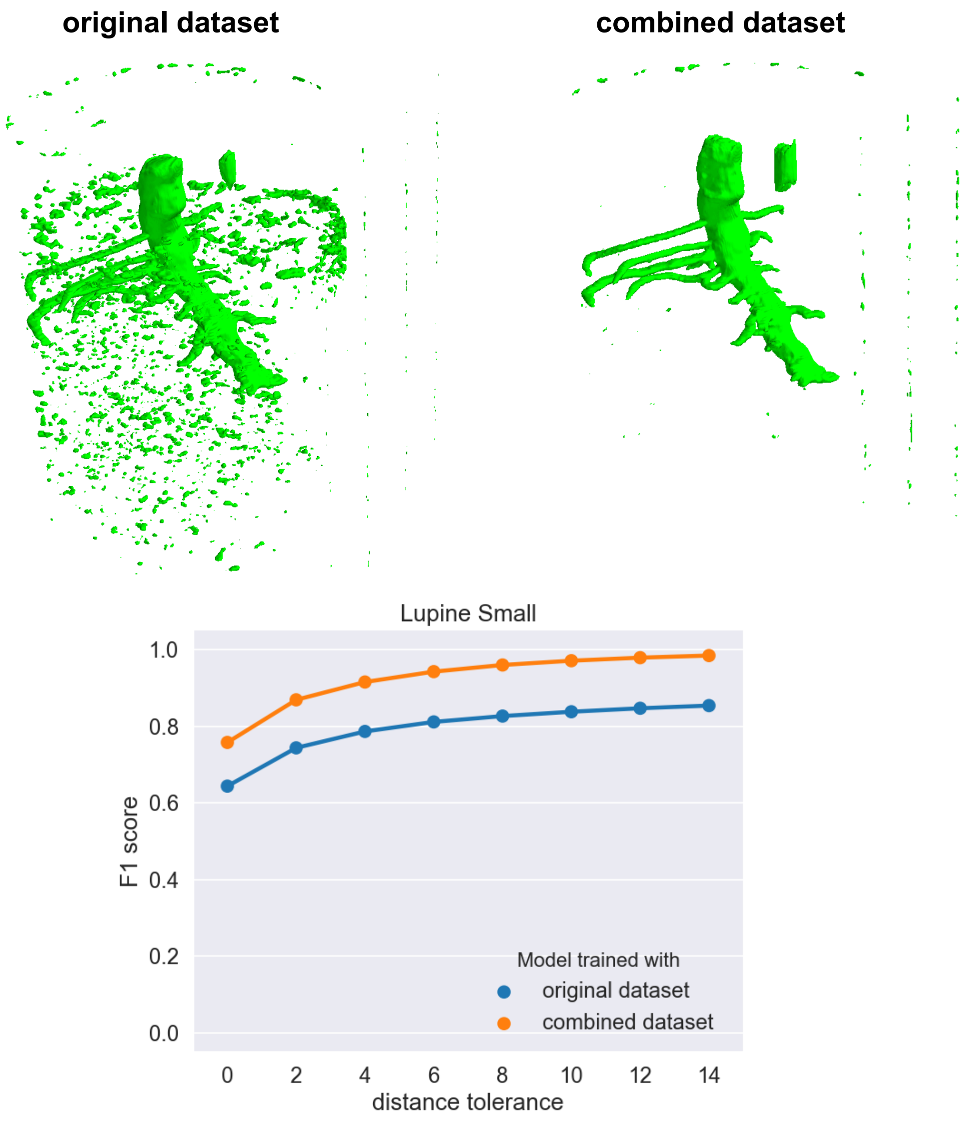


Figure 7.1: Comparison of segmentation results from models trained with the 2 different datasets, on the real Lupine Small data. The upper row shows the results thresholded at 0.5. The lower image compares the distance tolerant F1 scores of Model O (blue) and Model C (orange).

On the other hand, the results of real Lupine 22 show that Model C can produce a more connected root structure compared to Model O (Figure 7.2). The comparison between the distance tolerant F1 scores also shows a slightly improved performance of Model C (orange curve in Figure 7.2). This might be the benefit of integrating more diverse root structures in the dataset. Also, because the aliasing effect is incorporated in the randomly generated roots, the combined dataset mimics the thin roots in real images better.

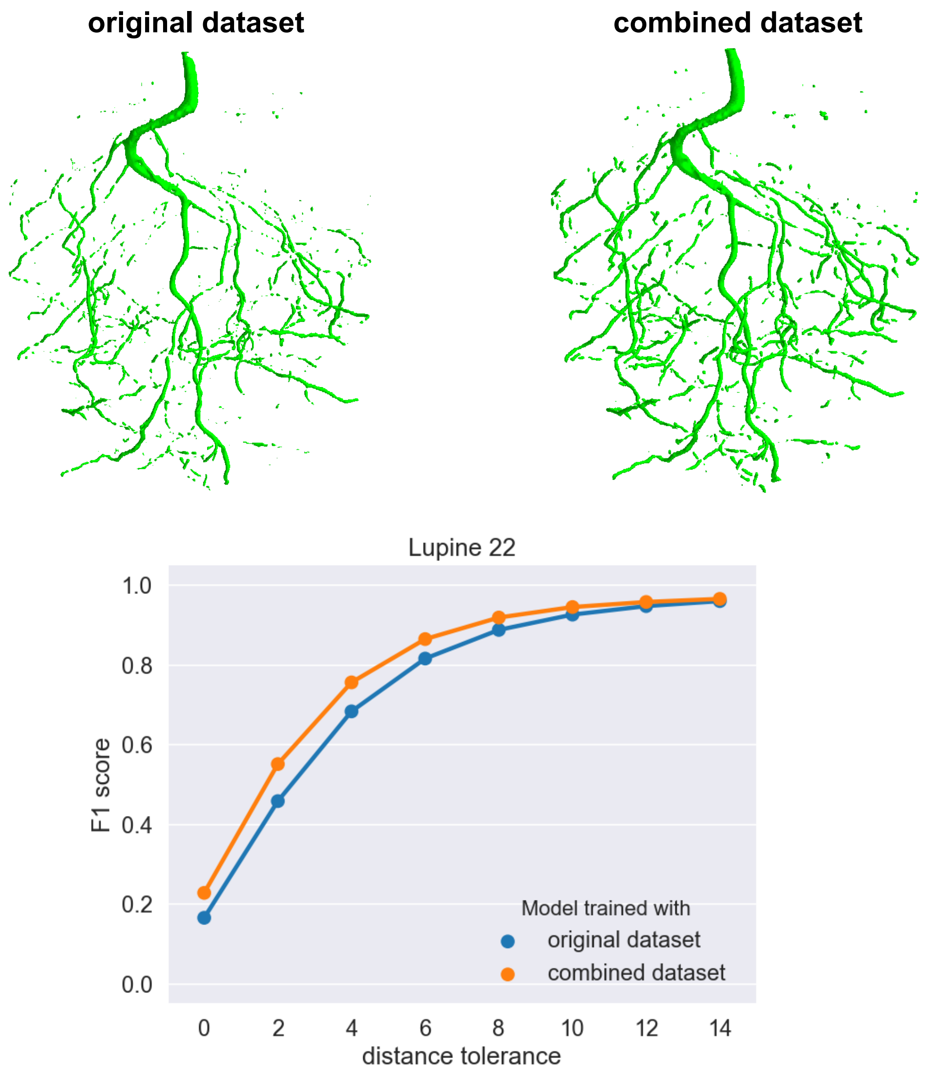


Figure 7.2: Comparison of segmentation results from models trained with the 2 different datasets, on the real Lupine 22 data. The upper row shows the results thresholded at 0.5. The lower image compares the distance tolerant F1 scores of Model O (blue) and Model C (orange).

For the overall comparison of the 2 models' performance, we plot the average distance tolerant F1 scores of all 5 real MRI images, under different distance tolerances (Figure 7.3). Because larger distance tolerance tolerates bigger misalignments between the segmentation result and the ground truth, the F1 score increases with the distance tolerance. Figure 7.3 shows that, the average test F1 score of Model C is higher than Model O, no matter what value the distance tolerance takes. This complies with the qualitative comparison between the visualized outputs above. In further experiments of this thesis, Model C will be used as the control model, if a different one is not explicitly mentioned.

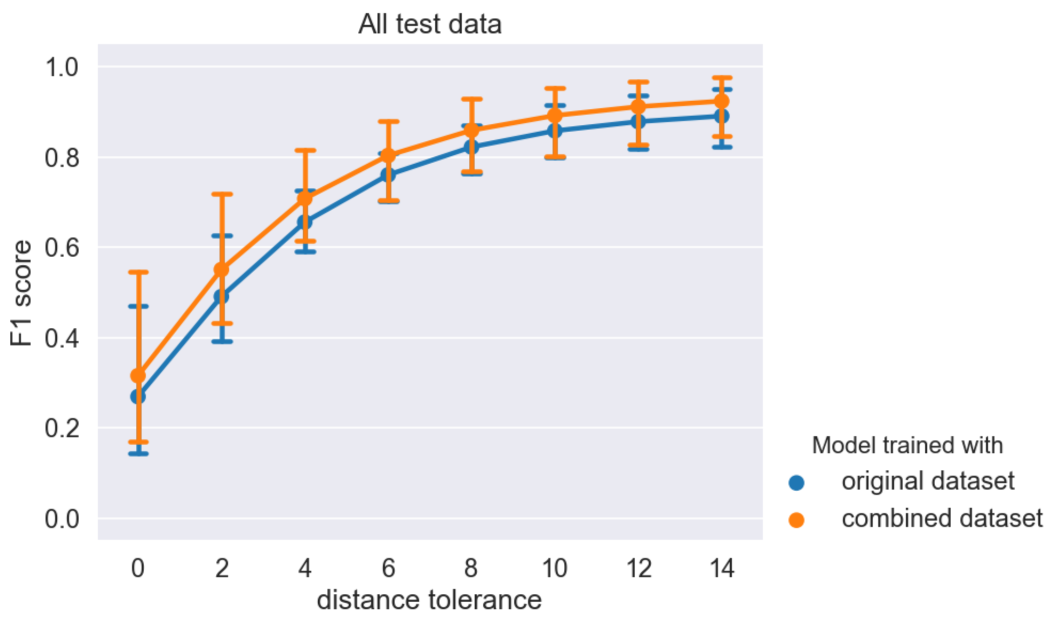


Figure 7.3: Comparison of distance tolerant F1 scores between Model O and Model C. Blue represents Model O, and orange represents Model C. The error bar shows the 95% confidence interval around the mean.

7.2 Effects of importance sampling

To speed up the learning convergence, importance sampling was used to train the network more frequently on the crops with more root voxels. In this experiment, the sampling probability is positively correlated with the percentage of root voxels in each crop. At the same time, we make sure that any crop has a non-zero probability of being sampled, by adding a small offset (0.02 is used here, see 6.2.1.2 for more details). Here we compare one model trained with uniform sampling and one with importance sampling in terms of the validation loss, to see which one converges first. The validation loss can be separated into 2 components: validation root loss and validation soil loss, which are the losses calculated from only the root voxels or only the soil voxels, respectively.

The validation soil losses of both the importance sampling model and the uniform sampling model are at the same level, but the validation root loss of the importance sampling model reaches a fast decreasing phase earlier than the uniform sampling model (Figure 7.4). The difference between the number of training batches needed to reach the plateau is around 5k, slightly more than one training epoch. This agrees with our expectation: because the importance sampling allows the model to learn from more root containing crops, the model can learn faster to detect root-like structures in the input, leading to faster convergence of validation root loss.

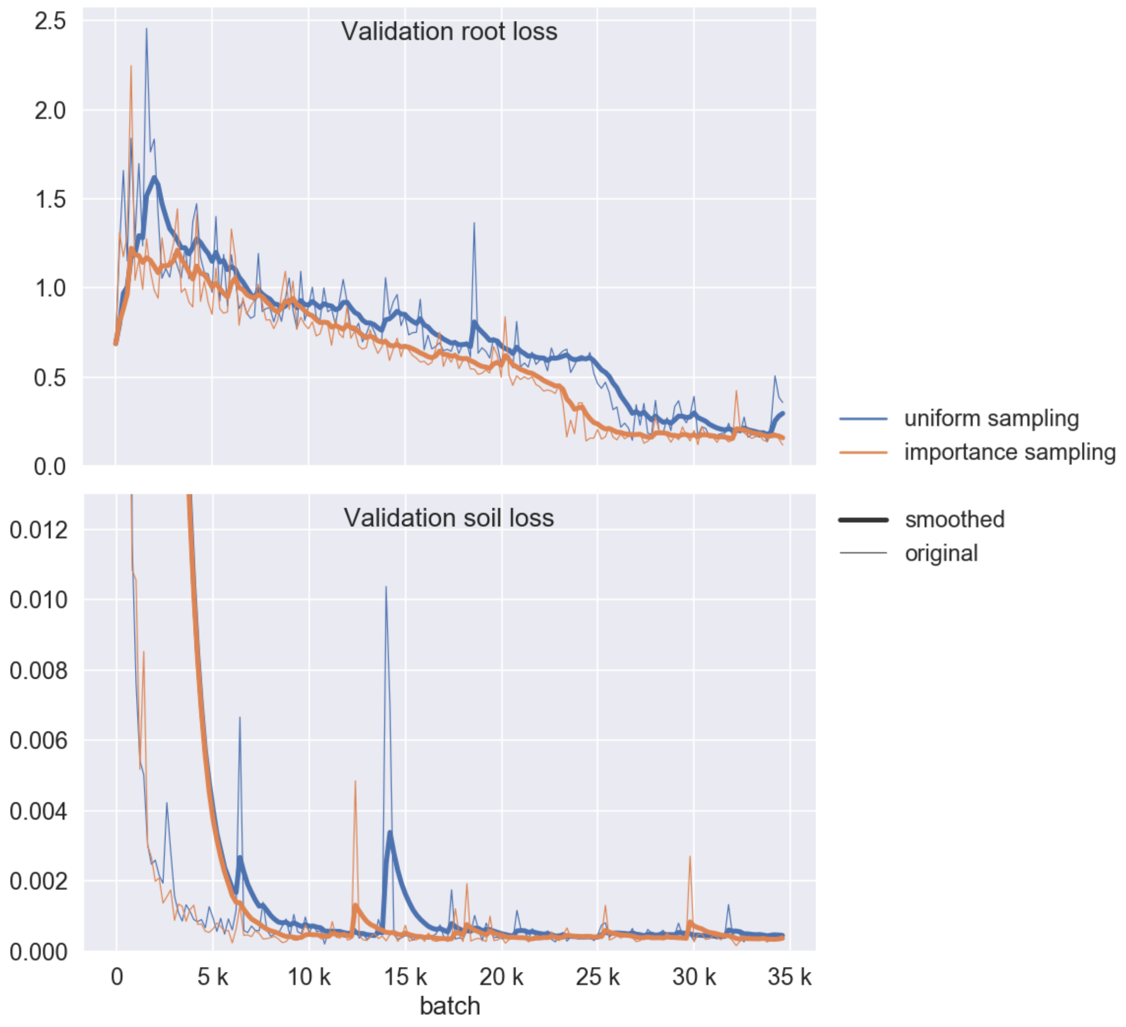


Figure 7.4: Comparison of the validation root loss curves between models trained with uniform sampling or importance sampling. The initial soil loss above 0.013 is clipped for a clearer presentation of the later training stage.

However, when we look at the distance tolerant F1 scores of these 2 models, there is no significant difference between them, no matter which distance tolerance value is used for computation (Figure 7.5). This can also be directly observed from the segmentation results on the test data, where there is no qualitative difference between these 2 models.

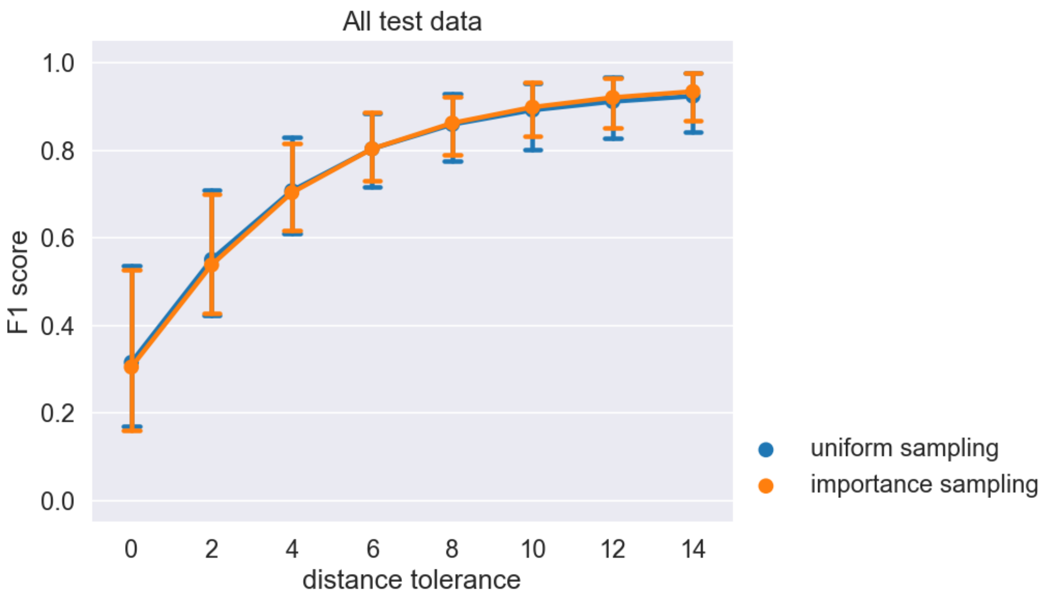


Figure 7.5: Comparison of distance tolerant F1 scores between the models trained with uniform sampling and importance sampling. Error bar shows the 95% confidence interval around the mean.

The result shows that importance sampling only speeds up learning convergence in terms of the number of training epochs, but does not improve the final performance on the test data. Moreover, the faster convergence speed is offset by the overheads of calculating the sampling probabilities for all possible crops in each image. This makes the actual time needed for the importance sampling model to converge even slightly longer. As a result, for further experiments, uniform sampling is used.

7.3 Effects of using don't-care flag

It is hard for the network to learn to accurately predict the root-soil border area. To make the model focus more on recognizing the main part of the roots instead of these unimportant details, we tried using the don't-care flag. It is used to label the root-soil border area, and voxels with this flag are ignored during training loss calculation. In what follows, we denote the model trained with the don't-care flag as the don't-care model.

The highest validation F1 score of the don't-care model is 0.993, higher than 0.964 of the control model (Table 7.1). However, this is because, for the don't-care model, the validation F1 is calculated only on voxels without the don't-care flag. These voxels are easier to segment, thus resulting in a higher F1 score. This is done to be consistent with the training process, for clearer observation of the learning convergence.

The result on the test dataset shows that when the distance tolerance is lower, the F1 score of the model trained with don't-care flag is slightly higher compared to the control model (Figure 7.6, upper plot). When the distance tolerance is high, this difference disappears. This increase of the F1 score comes from a higher recall and a slightly lower precision, which is less significant than the increase of the recall (Figure 7.6, lower plot). Therefore, applying the don't-care flag can indeed increase the recall of roots in the test dataset, which complies with our expectation.

When we look at the result of the real Lupine 22 data as one example, it can be observed that the root predictions by the don't-care model are significantly thicker than the control model. The proximate region of the root-soil border is more likely to be predicted as root by the don't-care model (Figure 7.7). The thickening of the root prediction can be explained as follows: The root prediction is slightly thicker than the ground truth, so the false negatives are likely to decrease. But the increased false positives will get ignored because they are most probably within the don't-care area. As a consequence, the network is penalized less. This root thickening effect is the most obvious in thinner roots such as Lupine 22, in which the root-soil border constitutes a larger proportion of the total root volume. Therefore, one reason why the recall is higher is that the thicker root prediction contains the ground truth inside it, thus the voxels of the root surface are less likely to be miss-predicted as non-root. Moreover, the predicted root structure of the don't-care model appears to be more connected than the control model, although several major disconnections in the control model result are still present in the don't-care model result. However, the price for thicker root predictions is that more non-root voxels in the vicinity of the root surface are also predicted as root, resulting in a lower precision.

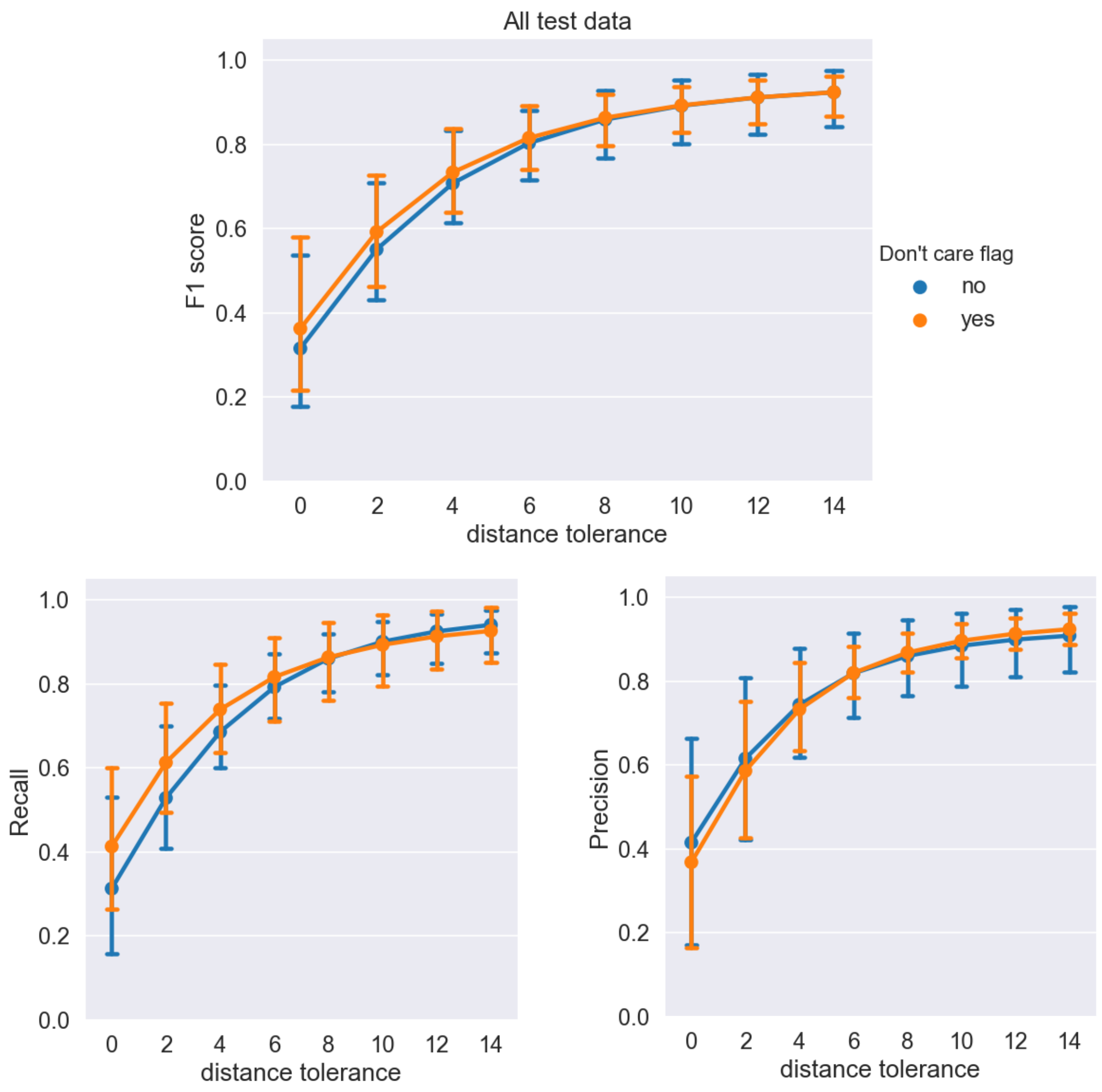


Figure 7.6: Comparison of evaluation metrics between the don't-care model and the control model. Error bar shows the 95% confidence interval around the mean.

Our original intention of using the don't-care flag is to increase the recall of roots so that some roots which were not detected before can be recognized. The actual results show that the increased recall is partly due to increased root thickness in the prediction, partly due to the correct detection of previously undetected roots. However, only some small gaps are bridged when applying don't-care mask. One drawback of using the don't-care flag is that the radii of the segmented roots are thicker than reality, which introduces an unwanted bias for the further root radius analysis.

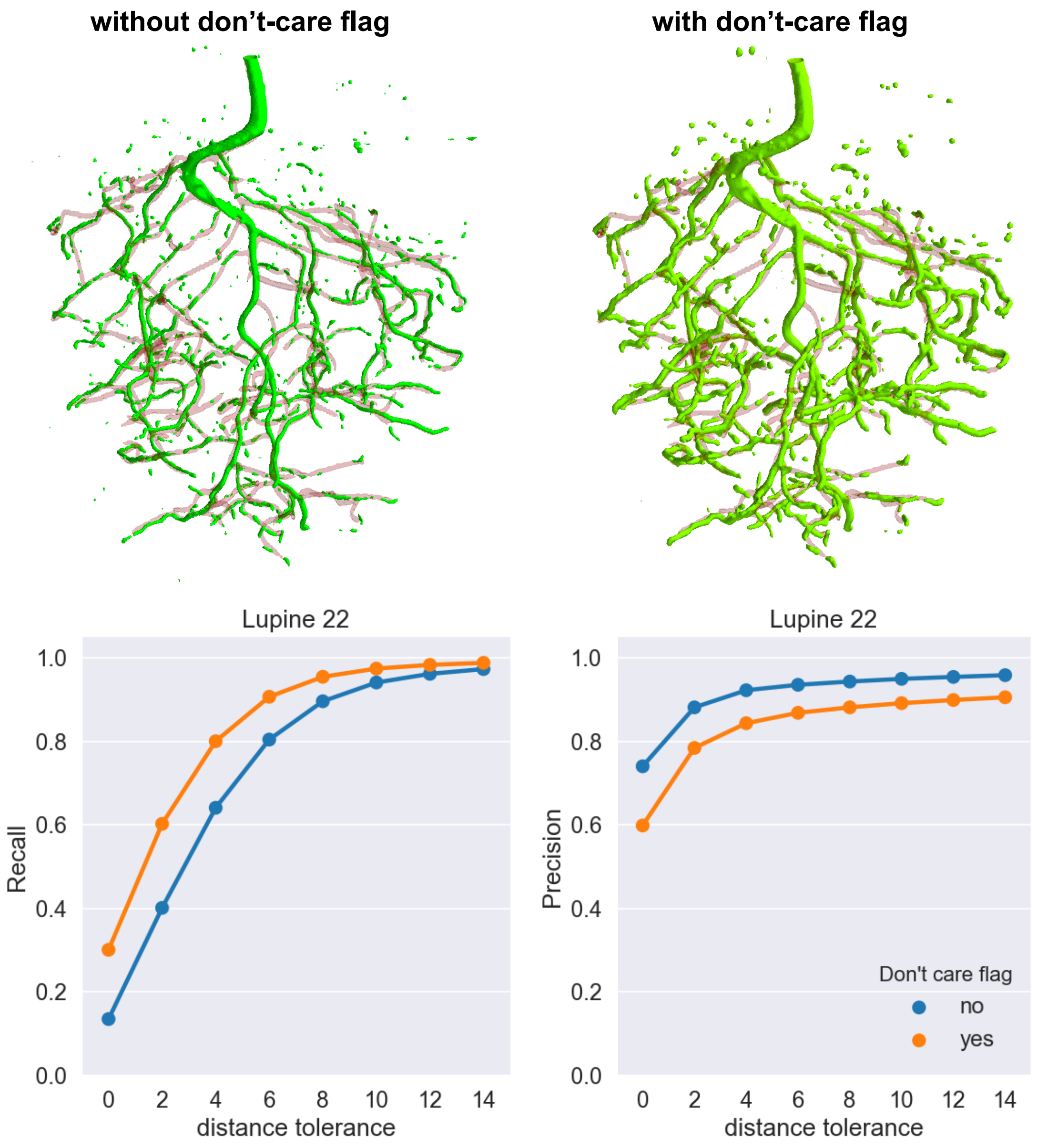


Figure 7.7: Comparison of the segmentation results on Lupine 22 between the don’t-care model and the control model. In the upper images, root predictions (green) are overlapped with the ground truths (red shade). The lower images are the comparisons of the distance tolerant recall and precision, respectively.

7.4 Effects of adding root at a later time point

Under the assumption that roots do not change their locations during growth but only increase the radii, the root at a later time point will contain all root voxels of an earlier time point in it. Therefore, one hypothesis is that adding the root image of a later time point may help the segmentation model confirm the detection of root voxels at an earlier time point. To test this hypothesis, we generated random virtual roots at 2 different time points, one earlier and one later, and combine with the real soil image to make them noisy. The noised roots of 2 different time points are then concatenated in the channel dimension, and used to train a 3D U-Net model (referred to as the multi-time model in the following). Another 3D U-Net model is trained with only the earlier roots for comparison (referred to as the control model in the following). The only difference between the model structures is that the first convolutional layers, which receive the input, have one more input channel for processing the root of the later time point.

The highest validation F1 score achieved by the multi-time model and the control model are 0.973 and 0.978, respectively. Both F1 scores are high, and the difference between them is not significant. One observation of the training process is that the validation loss of the multi-time model converges significantly faster than the control model (Figure 7.8). The difference in convergence time is roughly 6 epochs. This is probably because, with the help of the later root information, it is easier for the multi-time model to learn.

For testing, 6 randomly generated roots which are different from the training and validation ones are used. The real MRI images are not used here because we do not have the same roots imaged at a later time point. Although some new root data from our collaborator have different time points, they cannot be utilized because the roots at different time points have significant misalignments.

Because the test data here is virtual, there are no misalignments between the ground truth and the noisy image. Therefore, the F1 score on the test data can be calculated directly. The result shows that there is no significant difference between the 2 models in F1 scores as well as in recall (Figure 7.9). One observable difference is that the multi-time model result has slightly higher precision, which can be observed from the visualized output (Figure 7.10): for the output of the control model (left), there is a soil area detected as root (the

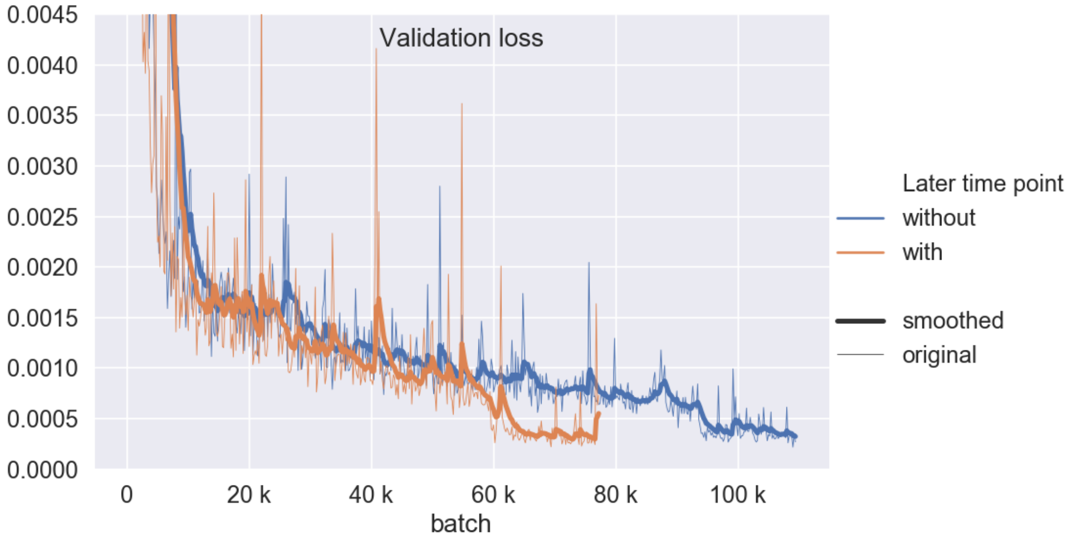


Figure 7.8: Comparison of the validation loss curves between the control model and the multi-time model.

top right part of image), but the output of the multi-time model does not have such a false positive prediction. This is probably because the network can infer from the image of the later time point which has a different noise pattern in the same area. When different noise patterns are present in the same area, the network may have a higher tendency towards predicting the voxels as non-root.

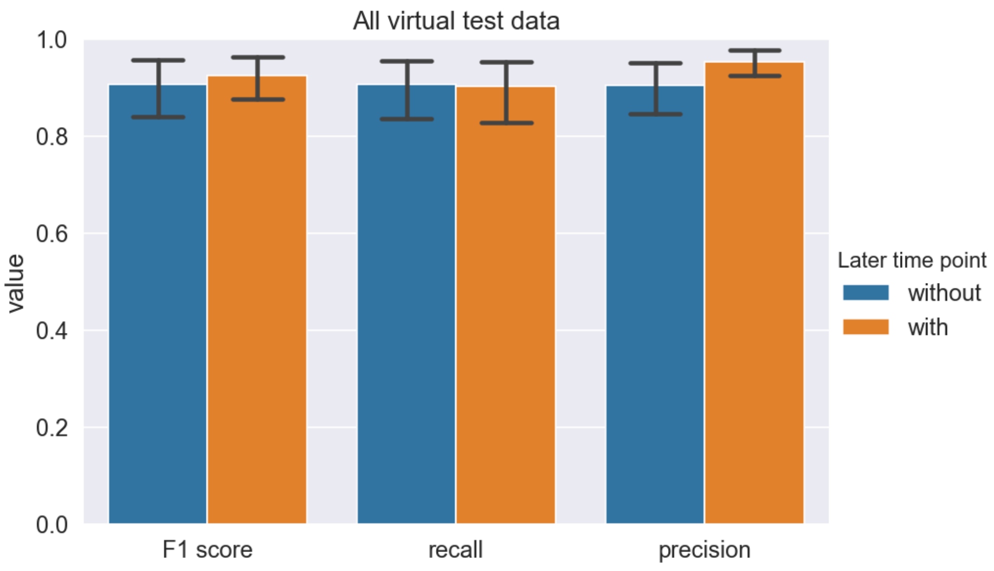


Figure 7.9: Comparisons of F1 score, recall, and precision, between the control model and the multi-time model.

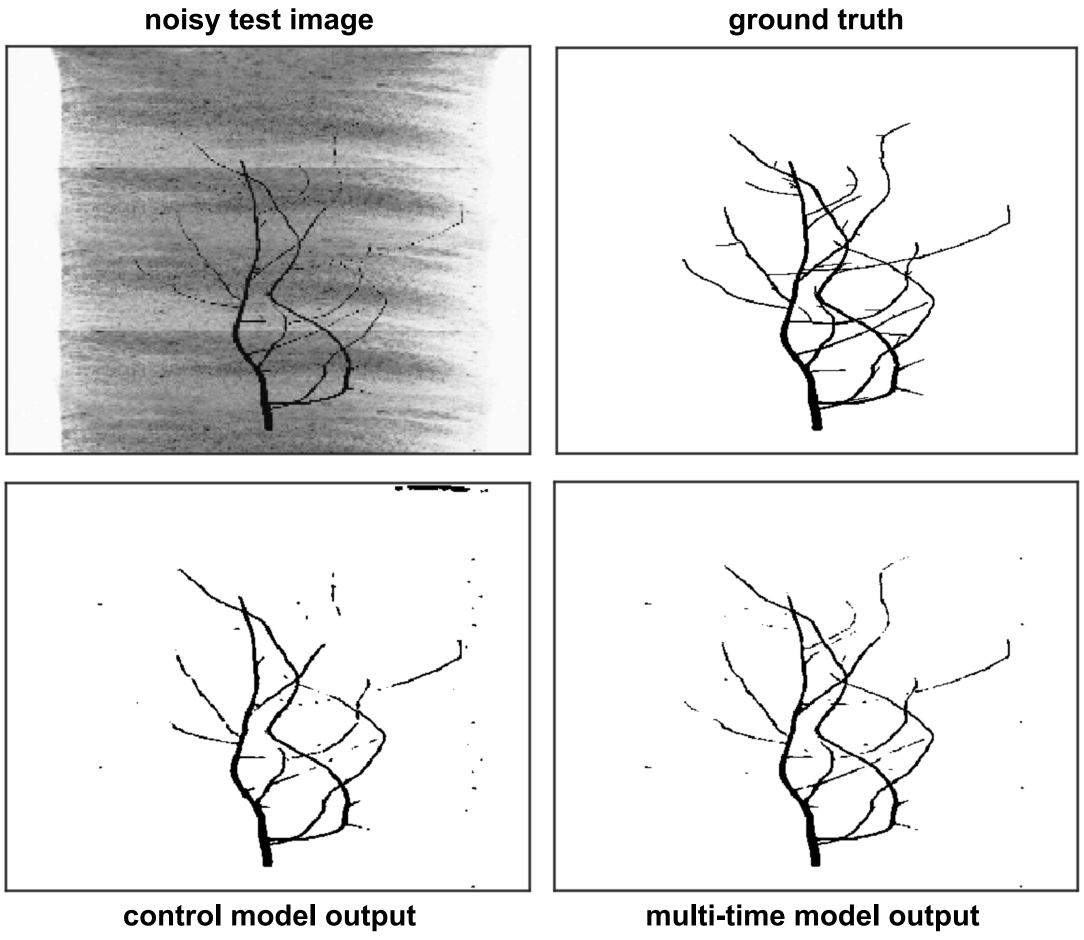


Figure 7.10: Comparison of the thresholded segmentation outputs on one virtual test image, between the control model and the multi-time model. The threshold applied to the segmentation is 0.5. The ground truth is shown on the upper right. Each of these images is the 2D maximum projection of the original 3D image along one horizontal dimension.

Moreover, although the average recall of the multi-time model is at the same level as the control model, the segmentation performance of the thinner roots seems better than the control model. For example in Figure 7.10, the thinnest root tips are more completely recovered in the multi-time model output. Overall, the segmentation result of the multi-time model appears to be finer and more detailed. However, because thinner roots contribute less to the recall compared to thicker roots, this improvement is not reflected in the overall recall value. Therefore, the results indicate that it is helpful for the network's performance to add the root at a later growth stage as the additional information.

As mentioned above, in order to reduce the disconnections in the detected roots, the aliasing effect is added to the randomly generated roots in the dataset. It can be observed from Figure 7.10 that, some root disconnections in the input image are indeed connected in both model outputs.

7.5 Effects of adding location-dependent information

From the observation of the real MRI images, we found some location information which may correlate with the existence of roots, and they may help improve the network's performance. Since these patterns are easy to compute, they can be provided directly as additional input channels for the network the learn from. Two different types of location-dependent information are tested individually. The same network structure is used as the control 3D U-Net model, except that the convolutional layers which directly receive the input have one more input channel, for processing the additional information channel.

7.5.1 Voxel-wise distance to the pot central axis

The voxel-wise distance to the pot central axis is used because of the observation that many root branches tend to grow along the pot border, away from the pot central axis. Meanwhile, the roots closer to the central axis grow almost horizontally in these real images. This distance information cannot be directly inferred from the input data by the network, because image crops instead of whole images are used as inputs. However, adding this new input channel did not change the highest validation F1 score significantly, which is approximately 0.964 for either model trained with or without this new channel (table 7.1). This may be partly due to the fact that the original validation F1 score is already high, which does not have much room for further improvement. The results on test data also show no significant difference after adding the new input channel (Figure 7.11). Also, no qualitative difference can be observed from the test segmentation outputs of the 2 models.

These results indicate that, at least in the current dataset, the distance to the central axis does not have enough correlation with the distribution of roots, thus is not able to provide much help in deciding if a voxel is root or non-root. But since the current dataset contains only 4 root structures from the real data

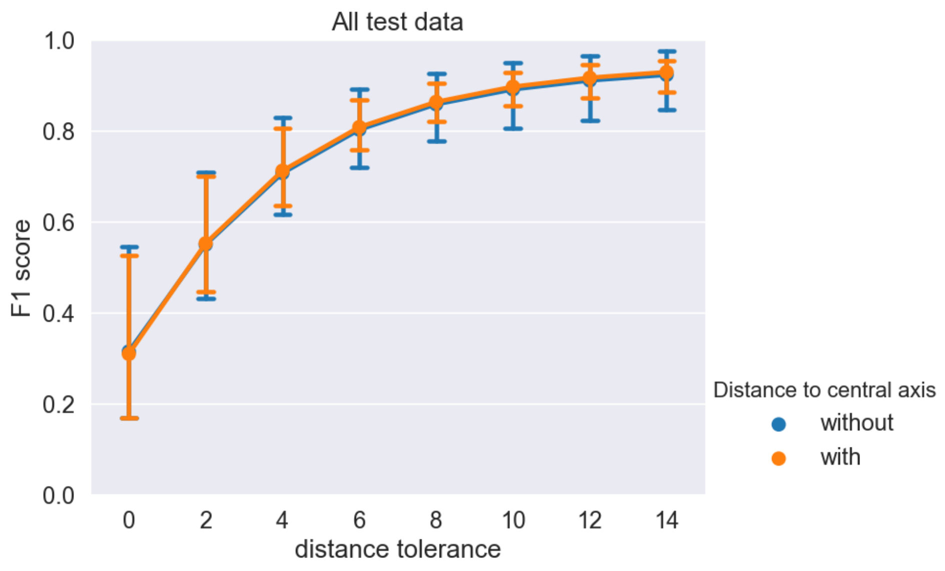


Figure 7.11: Comparison of distance tolerant F1 scores between the model trained with and without the voxel-wise distance to the pot central axis. Error bar shows the 95% confidence interval around the mean.

annotation and some randomly generated virtual roots, it may not represent the situation of the real roots so comprehensively. Therefore, in the future, we can try to imitate the observed relationships between this distance information and the root growth pattern in data generation. Or when it's possible to train with real data in the future, adding the distance information to the central axis is still worth trying.

7.5.2 Voxel-wise depth from the top of the pot

The voxel-wise depth information is used because we observed some features of the real root changes along the depth dimension. For example, the radii of roots tend to decrease and the roots at the top tend to grow horizontally until reaching the pot border.

The highest validation F1 score is 0.962 for the model trained with the depth information, which is similar to 0.964 of the control model (table 7.1). Furthermore, the distance tolerant F1 scores on test data show no significant difference from the control model trained without the additional information (Figure 7.12). These results suggest that adding the voxel depth information as input does not lead to improved segmentation performance. Similarly, the correlation between the depth and the root distribution is not deliberately incorporated in the training dataset, which may be the reason why the network couldn't make use of this additional information. Therefore, in the future, such features can be introduced in the random root generation for a new dataset. Or if we can use the real data for training, we should still try adding these location-dependent information channels then.

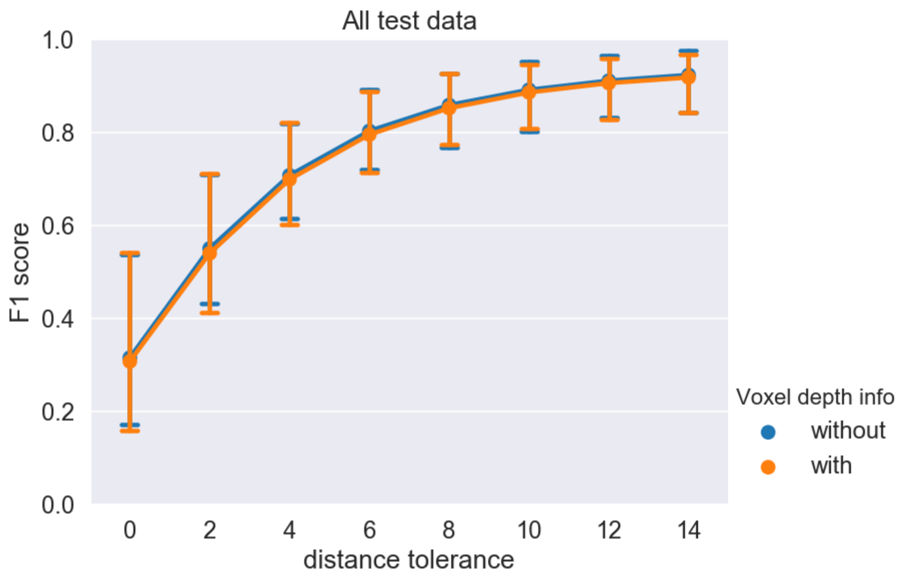


Figure 7.12: Comparison of distance tolerant F1 scores between the model trained with and without the voxel-wise depth information. Error bar shows the 95% confidence interval around the mean.

7.6 Effects of using a higher root weight

The dataset we use contains many times more non-root voxels than root voxels. To deal with this class imbalance, a value larger than 1 can be used to weight the predictions of the root voxels more heavily than the non-root voxels, in loss calculation. Here, we compare the segmentation performance of 2 models, one trained with a root weight of 10, the other the control model trained with root weight of 1. We observed that in the results on the real MRI images, the average F1 scores under different distance tolerances are similar for these 2 models. The tiny improvement of the model trained with the higher root weight is mainly due to the improved recall, with the cost of a slightly lower precision when the distance tolerance is low (Figure 7.13). It can be

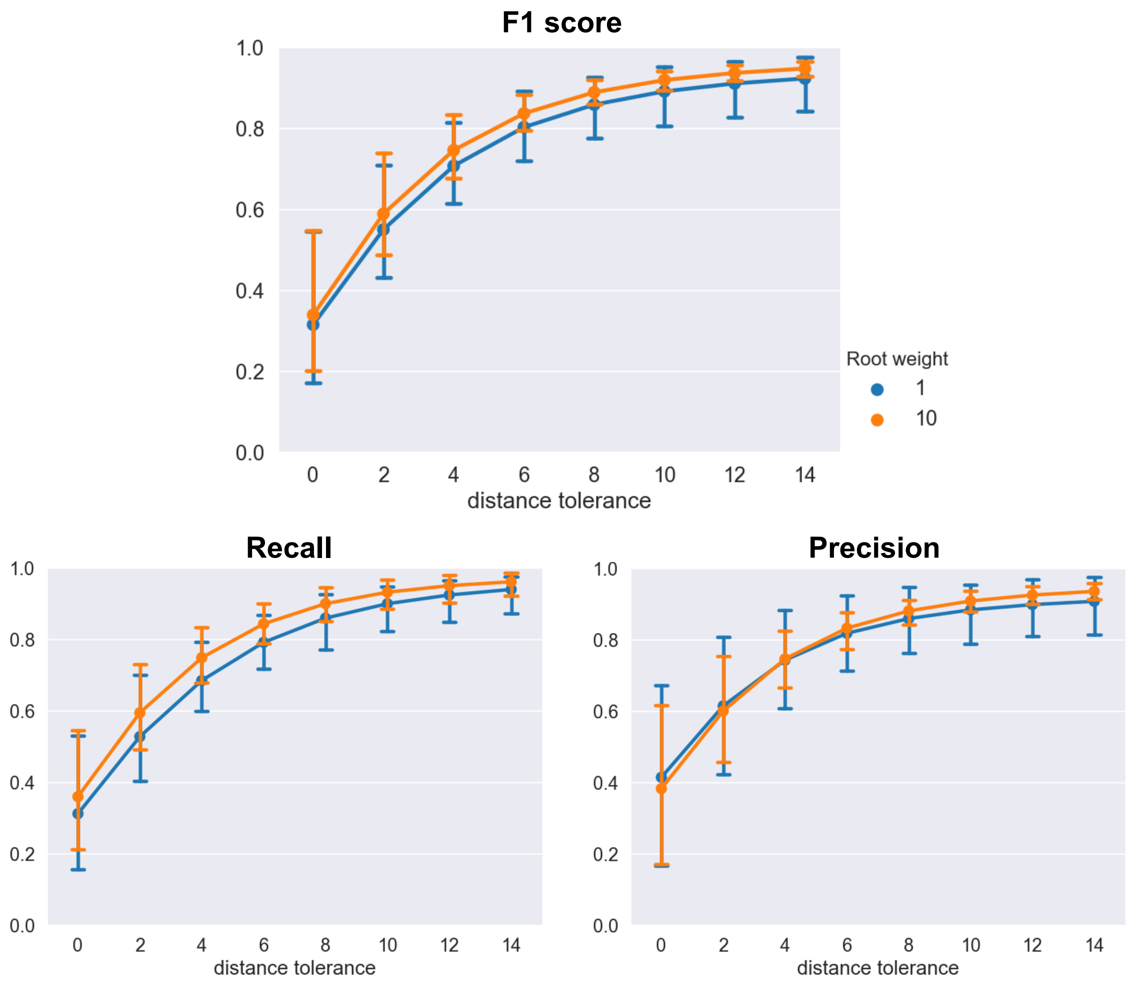


Figure 7.13: Comparison between the 3D U-Net model trained with a root weight of 10 and 1. Error bar shows the 95% confidence interval around the mean.

observed from the segmentation output of Lupine 22 that the root predictions are more connected but also more noisy (Figure 7.14). The explanation is that because the network gets punished more heavily when predicting a root voxel as non-root than the other way around, so it learns to favor the prediction as root when uncertain. This leads to a slightly increased recall.

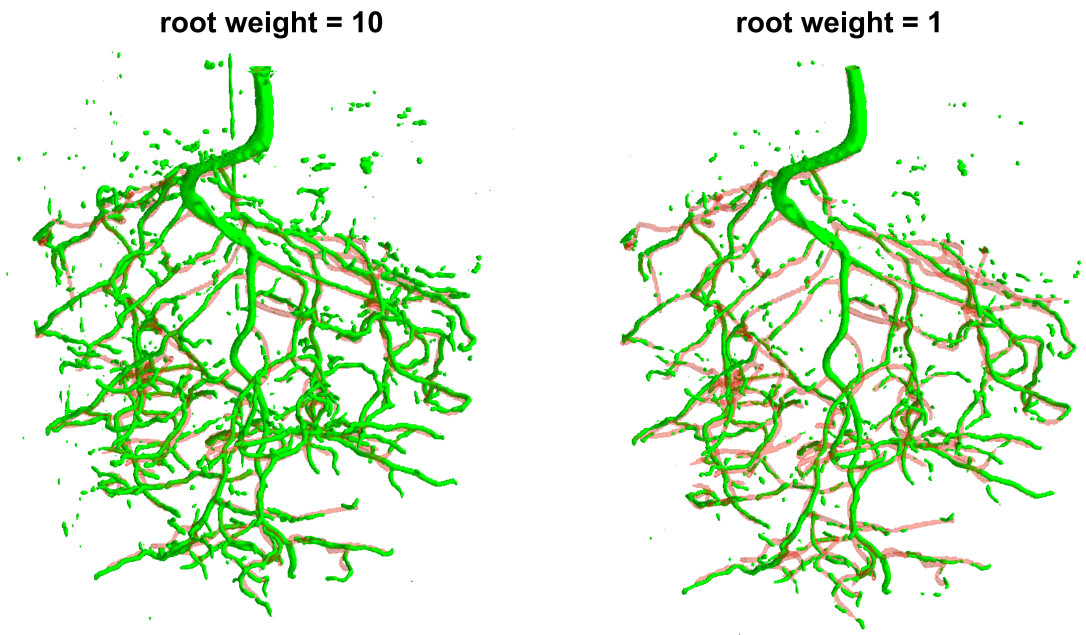


Figure 7.14: The thresholded segmentation outputs of the model trained with a root weight of 10 and 1, on the real Lupine 22 image.

7.7 Performance comparison with RefineNet

The segmentation performance of the 3D U-Net model is compared with the RefineNet model in a previous study (Uzman, Horn et al. 2019). The 3D U-Net model used for comparison is the same model as the one mentioned in Section 7.1, trained on the combined dataset. The performances are evaluated on the current test dataset of real MRI images, with the distance tolerant F1 score, recall, and precision (Figure 7.15). The F1 scores of the 2 models are similar when the distance tolerances are relatively low, but the F1 scores of the 3D U-Net model become slightly higher with increased the distance tolerance. This is mostly because the precision values of the 3D U-Net are higher than the RefineNet model while the recall values are lower, but the differences are not so big.

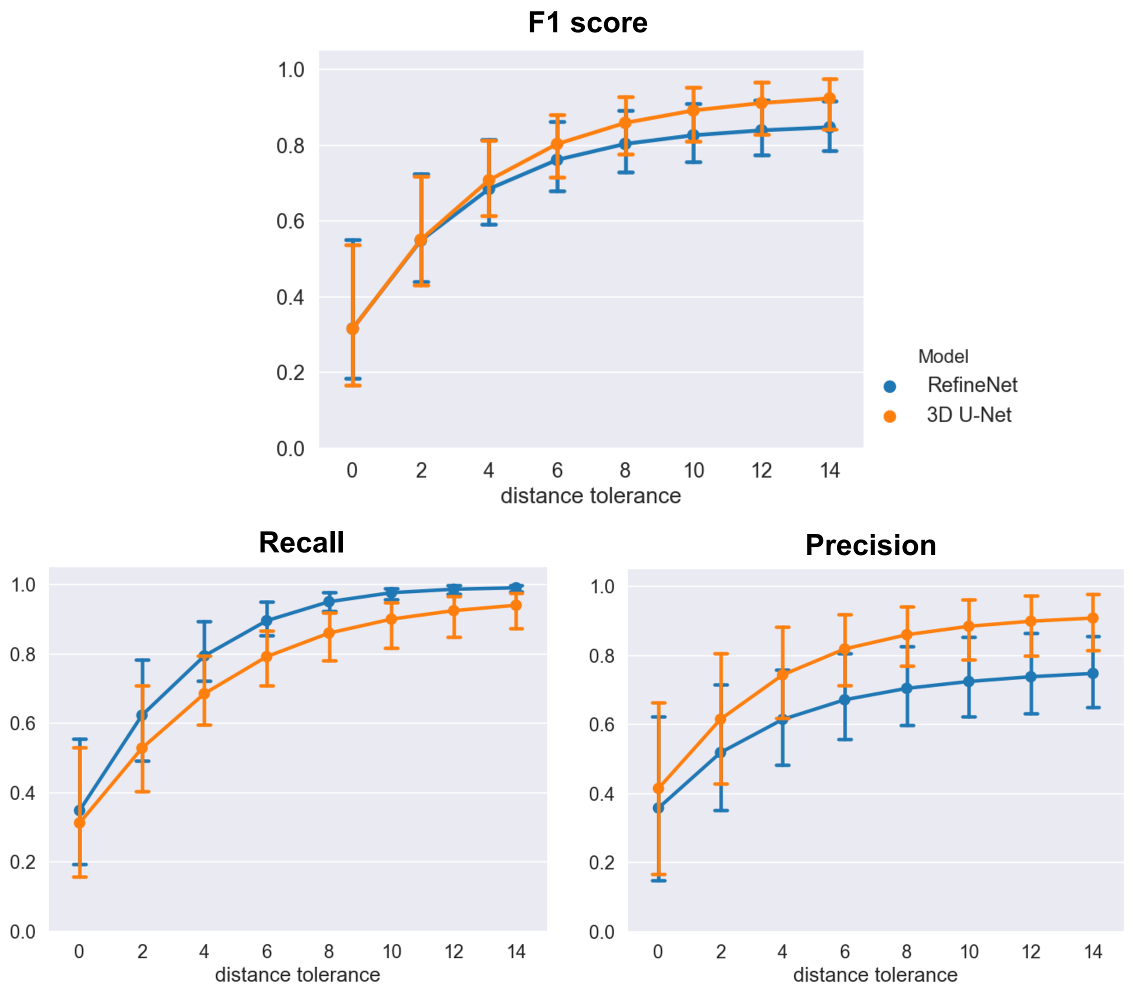


Figure 7.15: Comparison between the 3D U-Net model and the RefineNet model, in terms of the evaluation metrics. Error bar shows the 95% confidence interval around the mean.

These differences in recall and precision between the 2 models can also be observed directly from the segmentation outputs. For example, in the output of the Lupine 22 data from the RefineNet model, the predicted roots are more connected than that from the 3D U-Net model (Figure 7.16). This leads to a higher recall for the RefineNet model. However, there are also significantly more false-positive predictions in the non-root area compared to the cleaner output of the 3D U-Net model. Particularly, the planar MRI artifacts which are detected as roots by the RefineNet model are no longer recognized as roots in the 3D U-Net model, demonstrating the benefit of utilizing the depth dimension information. The disadvantage of having too many false positives is that the ones close to the root would confuse the root extraction algorithm, thus becoming incorporated into the extracted root structure. Meanwhile, it is still preferable to further increase the recall of the 3D U-Net output, since bridging wide gaps is also difficult for the root extraction algorithm.

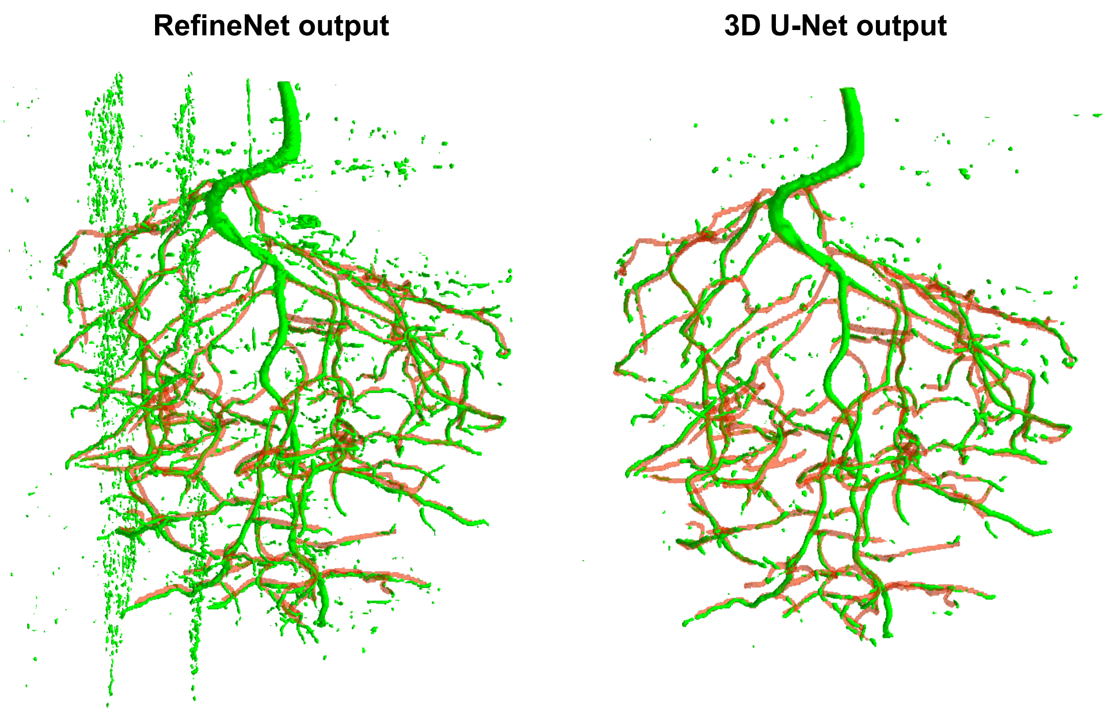


Figure 7.16: Comparison of the segmentation results on real Lupine 22 between the RefineNet model and the 3D U-Net model. The root predictions (green) are overlapped with the ground truth (red shade).

7.8 Root extraction results

Finally, we tried to extract the structural models from the segmentation results of the real MRI images, using a recently developed root extraction algorithm (Horn 2019) (Figure 7.17). Overall, the root structures can be decently extracted. The false positive predictions not so close to the root structure can be mostly ignored (marked with red in Figure 7.17), by adjusting certain parameters of the algorithm. It can be observed that when the gaps along root branches are small, they can be successfully bridged by the root extraction algorithm. But when the gaps are too big, either the algorithm fails to bridge them (e.g. the red root-like signals in Lupine 22, in Figure 7.17), or the bridging connections look unnatural (e.g. some blue connections in I_Sand_3D_DAP5, in Figure 7.17).

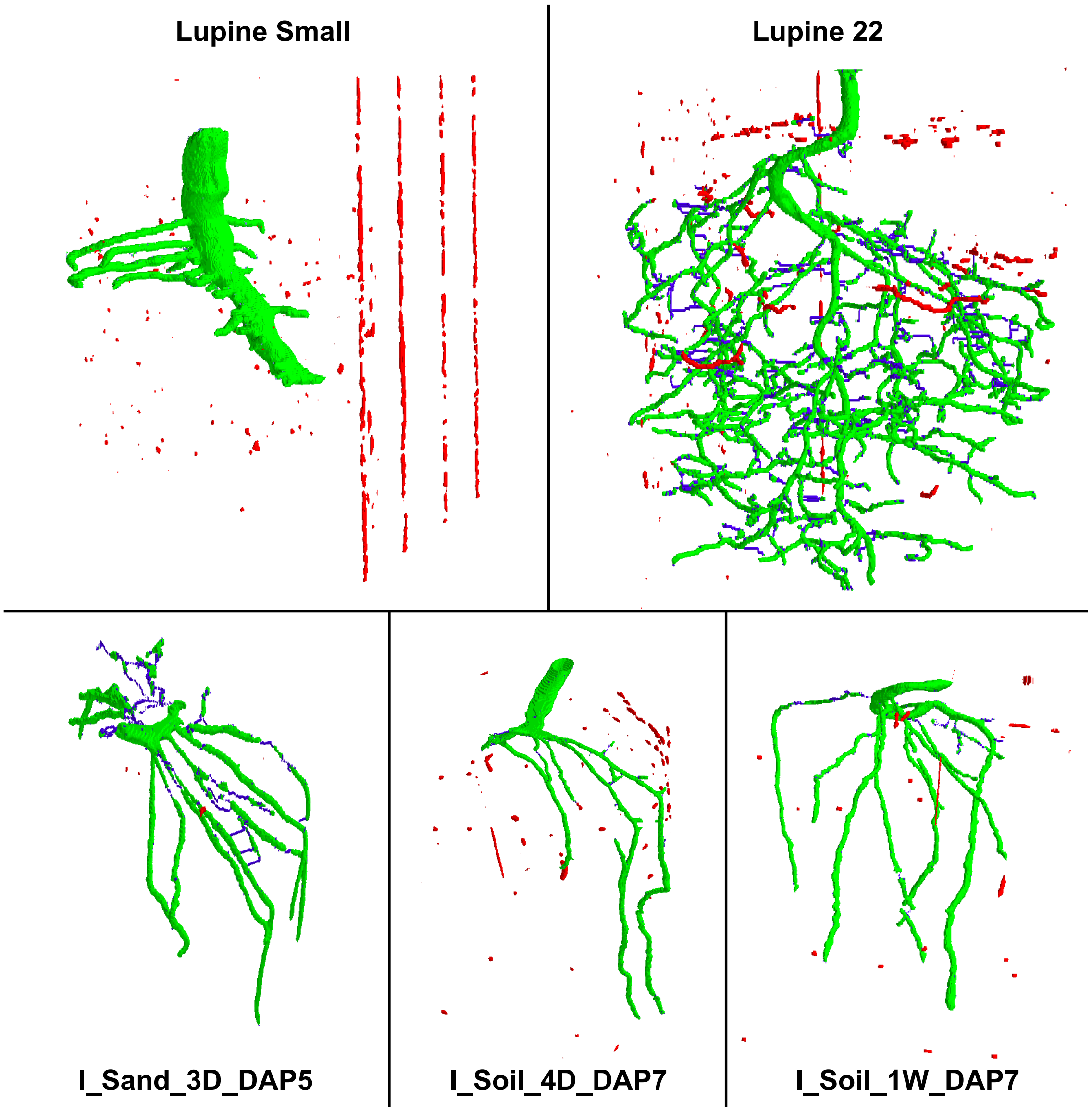


Figure 7.17: The results of structural model extraction on the 5 real MRI images. Green represents the overlapping between the extracted model and the segmentation result. Blue shows the gaps bridged in the extracted model. Red represents the positive predictions in the segmentation result that are ignored by the root extraction.

Model	**Highest validation F1 score**
Control, $rw = 1$	0.964
Importance sampling	0.962
Don't-care flag	0.993
Distance to the central axis	0.962
Voxel depth	0.964
$rw = 10$	0.944

Table 7.1: The highest validation F1 scores of each model that is validated on the validation set of the combined dataset. Here, rw represents the root weight used in loss calculation.

8 Conclusion

The thesis investigated semantic segmentation and super-resolution of plant root MRI images using a 3D U-Net model. The main contributions are as follows: Firstly, the use of 3D image crops as network inputs allows the use of deeper 3D CNNs, which is the 3D U-Net in our case. Furthermore, the 3D U-Net model generates better results with shorter training time, compared to the results of a previous study also using a 3D CNN (Horn 2018).

Moreover, the thesis has experimented with 3 types of additional input channels. When the root image at a later time point is used as the additional channel, the model's segmentation quality of thin roots is improved. However, there is no significant improvement in the evaluation metrics, which may be because the thin roots contribute less to the metrics than thicker roots. For the models that take location-dependent information as the additional input channel, we did not observe a significant enhancement of the segmentation performance. However, this may be because the generated dataset did not imitate the relationship between location-dependent information and root distribution well enough. Therefore, in the future, we can simulate this relationship in data generation, and then try these location-dependent information channels again.

Labeling the root-soil border area with don't-care flag leads to more connected root predictions, but the predictions are also thicker than in reality. This has the disadvantage that further root radius analysis may get biased by the thickening.

Applying importance sampling of image crops based on their root percentage indeed increased the learning convergence speed in terms of the number of training epochs, but did not increase the actual speed. This is because of the overheads of generating the sampling probabilities for all possible crops in each image.

Furthermore, we experimented with different root weights in loss calculation. Using a root weight larger than 1 leads to slightly higher recall of roots, but also a slightly higher amount of false positives. Therefore, we can adjust the balance between recall and precision as needed by adjusting the root weight value.

Another contribution is the improvement of the data generation. It is done by introducing more diverse root structures with a random root generation algorithm, simulation of the aliasing effect of thin roots, and the incorporation of real soil noise. When trained with this improved dataset, the model produces less false positives without sacrificing the recall in the segmentation results on real MRI images.

In the end, the 3D U-Net model is compared with the RefineNet model of another previous study (Uzman, Horn et al. 2019). Although the disconnections of the root predictions are more evident for the 3D U-Net, its results also contain a significantly lower amount of false positives. The false positives may be wrongly integrated into the root structure by the root extraction algorithm, so reducing the amount of them helps the algorithm extract a more accurate root structure. Besides, the disconnections of the 3D U-Net results can be reduced by setting a higher root weight, as long as the precision is constrained in a reasonable range. The results of root extraction on the segmentation results from our network show that small disconnections can be successfully bridged, but larger gaps are still problematic. Therefore, future work needs to further reduce root disconnections, but without increasing false positives, in order to achieve better root extraction results.

Bibliography

Abdel-Hamid, O., A.-r. Mohamed, H. Jiang and G. Penn (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. 2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP), IEEE.

Alain, G., A. Lamb, C. Sankar, A. Courville and Y. Bengio (2015). "Variance reduction in sgd by distributed importance sampling." arXiv preprint arXiv: 1511.06481.

Anand, R., K. G. Mehrotra, C. K. Mohan and S. Ranka (1993). "An improved algorithm for neural network classification of imbalanced training sets." IEEE Transactions on Neural Networks 4(6): 962-969.

Brown, R. W., Y.-C. N. Cheng, E. M. Haacke, M. R. Thompson and R. Venkatesan (2014). Magnetic resonance imaging: physical principles and sequence design, John Wiley & Sons.

Chawla, N. V., K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer (2002). "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16: 321-357.

Çiçek, Ö., A. Abdulkadir, S. S. Lienkamp, T. Brox and O. Ronneberger (2016). 3D U-Net: learning dense volumetric segmentation from sparse annotation. International conference on medical image computing and computer-assisted intervention, Springer.

Csurka, G., C. Dance, L. Fan, J. Willamowski and C. Bray (2004). Visual categorization with bags of keypoints. Workshop on statistical learning in computer vision, ECCV, Prague.

Dai, S., M. Han, W. Xu, Y. Wu, Y. Gong and A. K. Katsaggelos (2009). "Softcuts: a soft edge smoothness prior for color image super-resolution." IEEE Transactions on Image Processing 18(5): 969-981.

Dalal, N. and B. Triggs (2005). Histograms of oriented gradients for human detection.

Dong, C., C. C. Loy, K. He and X. Tang (2014). Learning a deep convolutional network for image super-resolution. European conference on computer vision, Springer.

Dong, C., C. C. Loy and X. Tang (2016). Accelerating the super-resolution convolutional neural network. European conference on computer vision, Springer.

Havaei, M., A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin and H. Larochelle (2017). "Brain tumor segmentation with deep neural networks." Medical image analysis 35: 18-31.

He, K., X. Zhang, S. Ren and J. Sun (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition.

Horn, J. (2018). "Superresolution 3D Image Segmentation for Plant Root MRI."

Horn, J. (2019). Private communications with Jannis Horn.

Hubel, D. H. and T. N. Wiesel (1962). "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex." The Journal of physiology 160(1): 106-154.

Ioffe, S. and C. Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv: 1502.03167.

Keys, R. (1981). "Cubic convolution interpolation for digital image processing." IEEE transactions on acoustics, speech, and signal processing 29(6): 1153-1160.

Kingma, D. P. and J. Ba (2014). "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980.

Krizhevsky, A., I. Sutskever and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems.

Lafferty, J., A. McCallum and F. C. Pereira (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data."

LeCun, Y., Y. Bengio and G. Hinton (2015). "Deep learning." nature 521(7553): 436-444.

Ledig, C., L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz and Z. Wang (2017). Photo-realistic single image super-resolution using a generative adversarial network. Proceedings of the IEEE conference on computer vision and pattern recognition.

Lin, G., F. Liu, A. Milan, C. Shen and I. Reid (2019). "Refinenet: Multi-path refinement networks for dense prediction." IEEE transactions on pattern analysis and machine intelligence.

Long, J., E. Shelhamer and T. Darrell (2015). Fully convolutional networks for semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition.

Loshchilov, I. and F. Hutter (2015). "Online batch selection for faster training of neural networks." arXiv preprint arXiv:1511.06343.

Maldonado-Bascón, S., S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gómez-Moreno and F. López-Ferreras (2007). "Road-sign detection and recognition based on support vector machines." IEEE transactions on intelligent transportation systems 8(2): 264-278.

Montufar, G. F., R. Pascanu, K. Cho and Y. Bengio (2014). On the number of linear regions of deep neural networks. Advances in neural information processing systems.

Moon, N., E. Bullitt, K. Van Leemput and G. Gerig (2002). Automatic brain and tumor segmentation. International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer.

Nagel, K. A., U. Schurr and A. Walter (2006). "Dynamics of root growth stimulation in Nicotiana tabacum in increasing light intensity." Plant, Cell & Environment 29(10): 1936-1945.

Nair, V. and G. E. Hinton (2010). Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10).

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer (2017). "Automatic differentiation in pytorch."

Perlin, K. (1985). "An image synthesizer." ACM Siggraph Computer Graphics 19(3): 287-296.

Pohlmeier, A., A. Oros-Peusquens, M. Javaux, M. Menzel, J. Vanderborght, J. Kaffanke, S. Romanzetti, J. Lindenmair, H. Vereecken and N. Shah (2008). "Changes in soil water content resulting from Ricinus root uptake monitored by magnetic resonance imaging." Vadose zone journal 7(3): 1010-1017.

Rahman, M. A. and Y. Wang (2016). Optimizing intersection-over-union in deep neural networks for image segmentation. International symposium on visual computing, Springer.

Rajan, D. and S. Chaudhuri (2002). "An MRF-based approach to generation of super-resolution images from blurred observations." Journal of Mathematical Imaging and Vision 16(1): 5-15.

Roth, H. R., L. Lu, A. Seff, K. M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey and R. M. Summers (2014). A new 2.5 D representation for lymph node detection using random sets of deep convolutional neural network observations. International conference on medical image computing and computer-assisted intervention, Springer.

Schnepf, A. and S. Behnke (2015). "Advancing structural-functional modelling of root growth and root-soil interactions based on automatic reconstruction of root systems from MRI."

Schulz, H., J. A. Postma, D. van Dusschoten, H. Scharr and S. Behnke (2013). Plant root system analysis from MRI images. Computer Vision, Imaging and Computer Graphics. Theory and Application, Springer: 411-425.

Schulz, H., J. A. Postma, D. Van Dusschoten, H. Scharr, S. Behnke, G. Csurka and J. Braz (2012). 3D Reconstruction of Plant Roots from MRI Images. VISAPP (2).

Sharma, V. (2018). "Deep Learning – Introduction to Convolutional Neural Networks." from https://vinodsblog.com/2018/10/15/everything-you-need-to-know-about-convolutional-neural-networks/.

Sun, Y., A. K. Wong and M. S. Kamel (2009). "Classification of imbalanced data: A review." International Journal of Pattern Recognition and Artificial Intelligence 23(04): 687-719.

Tahir, M. A., J. Kittler, K. Mikolajczyk and F. Yan (2009). A multiple expert approach to the class imbalance problem using inverse random under sampling. International Workshop on Multiple Classifier Systems, Springer.

Thai-Nghe, N., Z. Gantner and L. Schmidt-Thieme (2010). Cost-sensitive learning methods for imbalanced data. The 2010 International joint conference on neural networks (IJCNN), IEEE.

Thoma, M. (2016). "A survey of semantic segmentation." arXiv preprint arXiv:1602.06541.

Uzman, A. O., J. Horn and S. Behnke (2019). "Learning Super-resolution 3D Segmentation of Plant Root MRI Images from Few Examples." arXiv preprint arXiv:1903.06855.

van Dusschoten, D., R. Metzner, J. Kochs, J. A. Postma, D. Pflugfelder, J. Bühler, U. Schurr and S. Jahnke (2016). "Quantitative 3D analysis of plant roots growing in soil using magnetic resonance imaging." Plant physiology 170(3): 1176-1188.

Wang, Z., D. Liu, J. Yang, W. Han and T. Huang (2015). Deep networks for image super-resolution with sparse prior. Proceedings of the IEEE international conference on computer vision.

Wei, G.-Q., K. Arbter and G. Hirzinger (1997). Automatic tracking of laparoscopic instruments by color coding. CVRMed-MRCAS'97, Springer.

Yang, W., X. Zhang, Y. Tian, W. Wang, J.-H. Xue and Q. Liao (2019). "Deep learning for single image super-resolution: A brief review." IEEE Transactions on Multimedia.

Yu, F. and V. Koltun (2015). "Multi-scale context aggregation by dilated convolutions." arXiv preprint arXiv:1511.07122.