Rheinische
Friedrich-Wilhelms-Universität Bonn

Master thesis

Segmentation of Plant Root MRI Images

Author:
Ali Oguz Uzman

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Prof. Dr. Thomas Schultz

Advisor:
Prof. Dr. Sven Behnke

Submitted: 24.10.2018

Declaration of Authorship

I declare that the work presented here is original and the result of my own investigations. Formulations and ideas taken from other sources are cited as such. It has not been submitted, either in part or whole, for a degree at this or any other university.

Location, Date

Signature

Abstract

The plant roots have been a long standing research interest due to their crucial role for plants. As a non-invasive method, Magnetic Resonance Imaging (MRI) is used to overcome the opaque nature of soil and obtain 3D visualizations of plant roots. Existing algorithms fail to extract the structural model of the root when the environment (soil) is noisy and the resolution of MRI images is low. To this end, we develop a convolutional neural network to segment plant root MRI images as root vs non-root. The resulting segmentations have a higher resolution than their input MRI data.

Our convolutional neural network is based on RefineNet, a state of the art semantic segmentation method. As pretrained networks used in RefineNet expect 2D images, we use PCA to reduce 3D data into 2D RGB images for feature extraction. We test different loss functions to overcome the class imbalance problem between root and non-root voxels.

The provided data is insufficient for training a neural network. Thus, we develop data augmentation processes to create synthetic training data.

Our segmentation method is evaluated on both augmented and real data. For the real data, the ground truth data are not well aligned with the input MRI images, thus, we develop a metric which is robust against small misalignments between ground truth and our segmentations. The resulting segmentations of the plant root MRI images can successfully depict the root structures with some minor missing parts and MRI artifacts.

Contents

List of Figures

List of Algorithms

List of Tables

1 Introduction

The plant roots have been a long standing research interest due to their crucial role for plants (Dusschoten et al., 2016). Obtaining 3D visualization of these plant roots is possible with non-invasive methods such as X-Ray CT, neutron radiography or magnetic resonance imaging (MRI) which can overcome the opaque nature of the plant soil (Pflugfelder et al., 2017). Usually, the roots contain more water than the soils, this makes MRI a preferred way of obtaining these 3D visualizations by showing the local water content of the scanned plant root. Existing algorithms which extract the plant root structure from MRI root images fail when the environment (soil) is noisy and the resolution of the MRI data is too low to capture thin roots with precision (Schulz et al., 2012). This requires an external preprocessing step to reduce the noise and increase the resolution artificially.

This thesis is part of the project: *"Advancing structural-functional modelling of root growth and root-soil interactions based on automatic reconstruction of root systems from MRI"* (Schnepf and Behnke, 2015) . After acquisition of MRI images in the first part of the project, the second part involves reducing the noise found in these MRI root images. The third part of the project is to use the results of the second part to extract structural model of the plant roots.

This thesis addresses the second part of the project by segmenting the plant root MRI images as root vs non-root in super-resolution. In recent years, various machine learning methods have shown great success on many computer vision tasks such as image classification (Xie et al., 2017), object localization (Ren et al., 2015), action detection (Carreira and Zisserman, 2017), and semantic segmentation (Lin et al., 2017), etc. To this end, we make use of machine learning methods to learn root detection from noisy MRI images.

In this thesis, Convolutional Neural Networks (CNNs) are used to eliminate the noise by voxel-wise classification of the MRI images as root vs. non-root. The voxel-wise classifications are of higher resolution than their original MRI scan resolution, i.e, each input voxel must be mapped to multiple voxels in the estimated segmentation. We choose RefineNet for segmentation as it is a state of the art method for semantic segmentation. Pretrained networks used in RefineNet expect 2D RGB images. Thus, we map 3D data of ours into 2D RGB images for feature extraction.

Training these neural networks require large amounts of data. Due to current lack of sufficient input MRI scans and corresponding teacher data, we develop data augmentation processes to generate synthetic training samples.

This thesis is structured as follows: the problem is formally defined and the evaluation metrics are introduced in Chapter 2. Chapter 3 includes some basics on Convolutional Neural Networks and modern training methods. Some of the tasks and problems we have faced have been previously investigated by others, thus, we discuss the literature in Chapter 4 to build on our work. As we generate our own data for training and validation of the network, in Chapter 5, we describe the initial data we are given and the synthetic training data we generated. In Chapter 6, we introduce our method and discuss different parameterizations for training. In Chapter 7, we test the developed networks and different parameters described in Chapter 6. We finish this thesis in Chapter 8 with a conclusion and suggest future directions to extend the work.

Notation

$I^{x,y,z}$	Given a volumetric image I, the voxel of I that is centered at x, y, z.
$\mathbb{N}^0$	Set of all non-negative integers.
$\mathbb{R}^0$	Set of all non-negative real numbers.

2 Binary Segmentation

2.1 Binary Segmentation with Super-Resolution

Semantic segmentation can be defined as separating images into labelled regions. Usually, this is done on a pixel (in 2D images) or voxel (in 3D images) level where each voxel v of an image I is assigned to one of n predefined classes. To this end, most semantic segmentation algorithms output n values for each voxel or pixel v where $v_i \in [0, 1]$ refers to the confidence of the estimation algorithm, that v belongs to an object of class i. Then, v can be assigned to the class j with highest confidence among the others i.e. $v_j \geq v_k \; \forall \; k \in \mathbb{Z} : 0 \leq k < n$.

We refer to the special case of semantic segmentation where the number of classes $n = 2$ as *binary segmentation*. In this case, for each voxel v of an image I, a single confidence c can be interpreted as the probability of the said voxel belonging to the class 1. That is, given a certain threshold t, if $c \geq t$ then, the voxel v is assigned 1.

Aside from segmentation, another requirement of this project is to increase the resolution. Many roots have sub-voxel thickness. In original resolution, this may cause ambiguity in the occupancy of the voxels. To increase the resolution, we introduce super-resolution. We can formally define super-resolution as an estimation of a high resolution image from a lower resolution input (Hayat, 2017). This way, roots with sub-voxel thicknesses can be mapped to multiple supre-resolution voxels, allowing more precise segmentations. Then, the super-resolution binary segmentation can be defined as follows: for a super-resolution factor $k \in \mathbb{N}$, the function f must map from an input volumetric image

$$I \in \mathbb{R}^{x \times y \times z} \tag{2.1}$$

where $x \times y \times z$ is the resolution of I, to a super-resolution ground truth image:

$$G \in \mathbb{B}^{x' \times y' \times z'} : x' = x \cdot k, \; y' = y \cdot k, \; z' = z \cdot k. \tag{2.2}$$

Thus, each voxel of the input image is mapped to k^3 voxels in the estimated segmentation map. This thesis aims to find a super-resolution binary segmentation

function f such that

$$G = f(I). \tag{2.3}$$

To this end, we are going to use Convolutional Neural Networks to learn end-to-end mapping of low resolution noisy MRI images to their corresponding higher resolution ground truths.

2.2 Evaluation

Metrics are used to judge overall success o f segmentations. As only a small fraction of the MRI images is actually root, metrics which are robust against class imbalance are required.

We denote TP as the number of *true positives*, TN as the number of *true negatives*, FP as the number of *false positives*, FN as the number of *false negatives*.

2.2.1 F-Score

For binary classification tasks, F1-Score is a popular choice due its robustness against class imbalance. Its parameters are *precision* and *recall*.

$$precision = \frac{TP}{TP + FP}. \tag{2.4}$$

$$recall = \frac{TP}{TP + FN}. \tag{2.5}$$

In our case, high recall means most roots in ground truth are detected while high precision means most root predictions are matched with roots in the ground truth. F1-Score is the harmonic mean of precision and recall, this enforces both precision and recall to be high. It can be given as:

$$F1 = \frac{2 \cdot precision \cdot recall}{precison + recall}. \tag{2.6}$$

Alternatively, the F-Score can be calculated with different weightings of precision and recall. An example can be given as F2-Score. F2-Score weighs the recall more

than precision. It is defined as:

$$F2 = 5 \cdot \frac{precision \cdot recall}{4 \cdot precision + recall} \tag{2.7}$$

2.2.2 Distance Tolerant F-Score

As described later in section 5.2.1, the real MRI images and their provided ground truth are not well aligned. Thus, we introduce a new metric; *Distance Tolerant F1-Score* (Behnke, 2018) for robustness against small differences between MRI images and their provided ground truths. The intuition is to forgive false positives if they lie in a close proximity to the ground truth and false negatives if they lie in a close proximity to the prediction.

For a given distance tolerance, precision and recall are redefined. First, we define the volumetric dilation on volumetric images. Let $B \in \mathbb{B}^{x \times y \times z}$ and a distance tolerance $d \in \mathbb{R}$, then morphological dilation can be defined as:

$$D = dilate(B, d) \tag{2.8}$$

where,

$$D^{i,j,k} = \begin{cases} 1 : \exists\ \hat{i}, \hat{j}, \hat{k} :\ d \geq (|\hat{i}| + |\hat{j}| + |\hat{k}|) \text{ and } B^{i+\hat{i},j+\hat{j},k+\hat{k}} = 1 \\ 0 : \text{otherwise} \end{cases} \tag{2.9}$$

In equation 2.8, D is dilation of B by d. Equation 2.9 implies that, for a voxel v of B at location x, y, z, if there exists a voxel u whose Manhattan distance to v is less than d and the value of u is 1, then, $D^{x,y,z} = 1$.

Let G be the ground truth and S be the predicted segmentation, we define dilated ground truth $\hat{G}$ and dilated predicted segmentation $\hat{S}$ as;

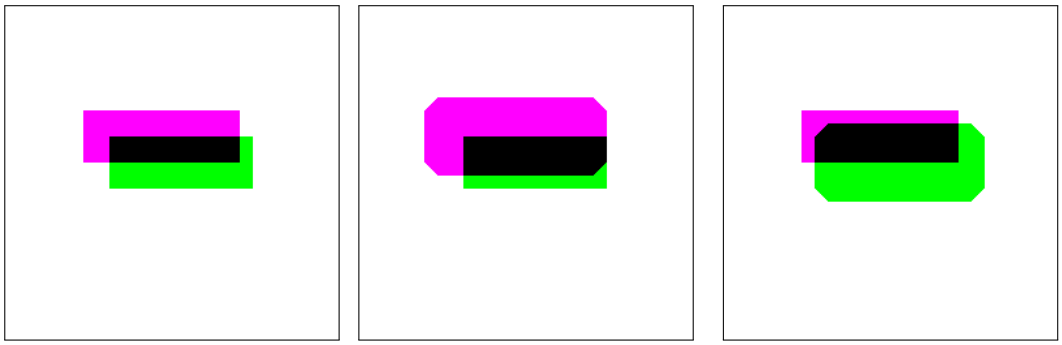
$$\hat{G} = dilate(G, d), \tag{2.10}$$

$$\hat{S} = dilate(S, d). \tag{2.11}$$

The distance tolerant precision is defined as;

$$precision' = \frac{\sum_{i,j,k=0} \hat{G}^{i,j,k} \cdot S^{i,j,k}}{\sum_{i,j,k=0} S^{i,j,k}} \tag{2.12}$$

Equation 2.12 implies that when calculating the precision, for each positive prediction $s \in S$, if there exists a positive ground truth voxel $p \in G$ and $distance(p, s) \leq d$,



(a) Original segmentation and ground truth **(b)** Dilated ground truth **(c)** Dilated segmentation

Figure 2.1: In (a), a misalignment between ground truth and segmentation prediction is shown. Purple areas are the false negatives, green areas are the false positives, and black areas are the true positives. Let us assume that the mismatch between these two occurs due to an annotation mistake in the ground truth. To compensate for such a misalignment, distance tolerant F1-Score is used. In (b), ground truth is dilated by 10 and distance tolerant precision is calculated. In (c), predicted segmentation is dilated by 10 and distance tolerant recall is calculated.
(a) Precision: 0.45, Recall: 0.42, F1-Score: 0.43
(b) Precision: 0.73, (c) Recall=0.68, Distance tolerant F1-Score: 0.71

then consider s a true positive.

$$recall' = \frac{\sum_{i,j,k=0} G^{i,j,k} \cdot \hat{S}^{i,j,k}}{\sum_{i,j,k=0} G^{i,j,k}} \tag{2.13}$$

Similar to equation 2.12, equation 2.13 implies that when calculating the recall, for each positive ground truth voxel $p \in G$, if there exists a positive prediction $s \in S$ and $distance(p, s) \leq d$, then consider p a true positive.

Now, distance tolerant F1-Score can be redefined as,

$$F1' = \frac{2 \cdot precision' \cdot recall'}{precision' + recall'}. \tag{2.14}$$

Theoretically, as d increases, both precision, recall, and F1-Score should be converging to 1 regardless of the actual quality of the segmentation. Thus, it is important to select a distance threshold that is reasonably small and that does not blow up the precision and recall giving an unrealistically high F1-Score.

2.2.3 Evaluation Criteria

The difficulty of segmenting the MRI images depends on the signal-to-noise ratio (SNR). Higher SNR means higher root intensity and lower noise, thus, segmentation is easier. The definition of SNR is given with more details in section 5.3.

To estimate the quality of the segmentations, the F1-Scores with respect to different SNR levels are investigated. For an SNR level of i, true positives $\hat{TP}$, false negatives $\hat{FN}$, and false positives $\hat{FP}$ are defined as the sum of the number of true positives, false negatives, and false positives in all MRI images of SNR level i respectively. The overall F1-Score for SNR level i is calculated with the equations 2.4, 2.5, and 2.6 using $\hat{TP}$, $\hat{FN}$, and $\hat{FP}$.

For a final average F1-Score among the whole validation set V, where $v \in V$ is a ground truth-input MRI root image pair; the average F1-Score is calculated as follows:

$$avg\_f1 = \frac{1}{|V|} \sum_{v \in V} F1Score(v). \tag{2.15}$$

This value is used to compare the accuracy of our method with the method of Horn, (2018).

3 Theoretical Background

3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a type of neural networks specialized for grid-like inputs (Goodfellow, Bengio, et al., 2016). Mostly used for 2D images, the assumption is that the pixels that are close to each other are more related than the ones that are far away.

3.1.1 3D Convolutional Neural Networks

Plant root MRI scans are 3D images. Although it may sound reasonable to use 3D convolutional networks with 3D data, 3D CNNs have known issues. While modern deep learning frameworks such as TensorFlow (Abadi et al., 2016), and PyTorch (Paszke et al., 2017) support 3D convolutions, their use has been limited due to higher memory and computational time requirements compared to their 2D counterparts (Horn, 2018). At the time of the writing of this thesis, 12GB frame buffer is the highest memory available for modern GPUs (e.g. NVIDIA Titan X). However, Horn, (2018) has shown that even shallow 3D CNNs run into memory problems. While training is time consuming due to 3D convolutions, the bigger issue is the memory consumption of the 3D activations and kernels. This limits the possibility of implementing deeper architectures with wider layers. This memory constraint can be overcome by splitting the input images to 3D patches, however, this increases the training time even further.

3.1.2 Activation Functions

Most classification problems are not linearly separable. Since convolution operation can only express linear functions, this limits the networks to learning only linear functions. Activation functions are applied after these linear operations to introduce non-linearity to the networks. Selection of activation function is crucial for the network performance as investigated by several researchers (e.g. Glorot and Bengio, (2010)). We present two of the relevant ones used in our work.

Sigmoid

Sigmoid is defined as,

$$\sigma(x) = \frac{1}{1 + exp(-x)}, \tag{3.1}$$

and its derivative,

$$\frac{d}{dx}\sigma(x) = \sigma(x) * (1 - \sigma(x)). \tag{3.2}$$

Sigmoid function has a range [0,1], making it suitable to represent probabilities and confidences. However, its derivative is too small especially when $\sigma(x) \approx 0$ or $\sigma(x) \approx 1$. Thus, when used in consecutive layers, it causes saturation of the learning process (Glorot and Bengio, 2010).

Rectified Linear Unit

Rectified linear unit (ReLU) is defined as,

$$ReLU(x) = max(x, 0), \tag{3.3}$$

and its derivative,

$$\frac{d}{dx}ReLU(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & otherwise \end{cases} \tag{3.4}$$

ReLU is widely used in recent successful deep learning architectures. Its derivative is 1 when $x > 0$; enabling the efficient flow of gradients backwards through the network.

3.2 Adam Optimizer

While stochastic gradient descent (SGD) based algorithms have become the norm to train the neural networks, pure SGD comes with stabilization issues and difficulty of choosing optimal hyper-parameters. Success of the training heavily relies on the selection of a correct learning rate and decaying it correctly over time (Schaul et al., 2013). Several methods such as *momentum* have been proposed to help SGD learn more efficiently, yet hyper-parameter selection remains a problem. Kingma and Ba, (2014) propose *Adam* to improve SGD by adjusting the learning rate automatically. Adam keeps separate learning rate for each parameter and

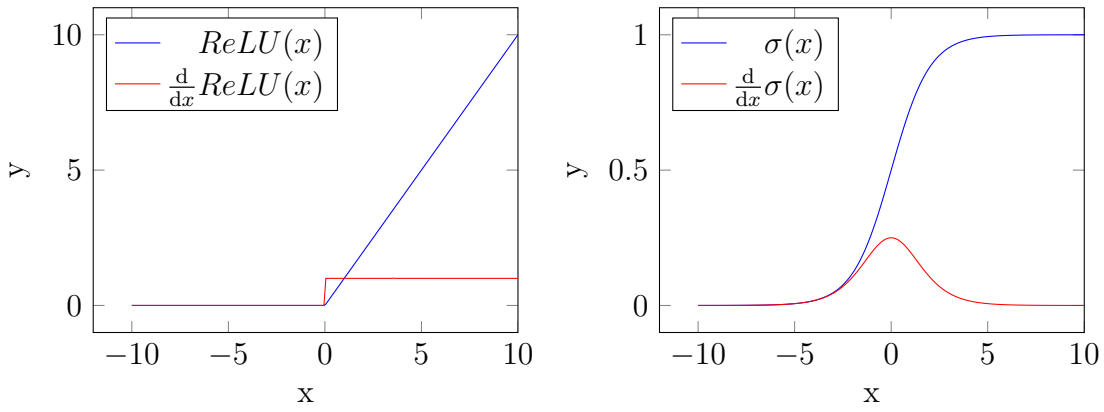


Figure 3.1: ReLU, Logistic Sigmoid and their derivatives.

handles the momentum internally which reduces the need to carefully adjust the hyper-parameters. As studies by the authors show, Adam usually outperforms SGD and other SGD based algorithms (Kingma and Ba, 2014).

4 Related Work

4.1 Semantic Segmentation

Semantic segmentation is an extensively researched area with different approaches. Currently, the state of the art results are obtained using deep learning methods.

Previously, methods such as random forests and Support Vector Machines (SVMs) have been successfully applied to semantic segmentation tasks. For example, Kolesnik and Fexa, (2004) train SVMs for segmentation of wounds from RGB images. Another example uses random forests; Schroff et al., (2008) train several binary trees using randomly sampled features, which are aggregated to make a final decision for each pixel.

These approaches are old and have been outperformed using more modern techniques. As with many of the computer vision tasks, deep learning methods achieve state of the art results for semantic segmentation. Usually, these methods involve either encoder-decoder architectures or transfer learning techniques.

Fully Convolutional Networks (Long et al., 2015), successfully utilized the first end-to-end, supervised training for semantic segmentation using only convolutional layers. The network adopts an encoder-decoder architecture; through a series of convolution and pooling layers, the image resolution is reduced to a latent space. This is followed by a series of transposed convolution layers which learn to upsample the image to the input size. Another example is U-Net (Ronneberger et al., 2015). Based on FCNN architecture, U-Net (Ronneberger et al., 2015) concatenates the activations of the encoder of the network to the activations of the decoder. This way, more precise segmentations are obtained.

Instead of using pooling layers, DeepLab (L.-C. Chen et al., 2017) incorporates dilated convolutions which increase the receptive field of the convolutional layers and keeps the same computational time complexity. DeepLab achieves some of the best results for common semantic segmentation datasets.

Lin et al., (2017) introduced RefineNet which produces high resolution segmentations by iteratively refining lower resolution activations of ResNet (He et al., 2016) to higher resolutions. The unique structural properties of RefineNet allows the construction of high resolution semantic segmentation networks easily.

In medical field, semantic segmentation methods are frequently used to detect tumors from brain MRI images. Although the 3D CNNs have been in in use for some time, due to their high memory and computational power requirements, 2D CNNs are usually preferred over their 3D counterparts. For example, Pereira et al., (2016) approach brain tumor segmentation as a problem with 5 classes (normal tissue, necrosis, edema, non-enhancing, and enhancing tumor) using 2D convolutions.

Zikic et al., (2014) present another 2D method for segmentation of brain tumors by treating the depth along one axis as feature channel, which enables the use of simple 2D convolutions. This is an idea that we adopt for our method.

A 3D CNN example is given by Kleesiek et al., (2016) who train a network with 8 layers for skull stripping from MRI images, achieving state of the art results.

4.2 Super-Resolution

Deep learning methods can learn super-resolution by end-to-end mapping of low resolution images to high resolution. Most of these methods only aim to increase the resolution rather than segmenting the image. To this end, self-supervised learning is utilized. An input image is downscaled to a lower resolution. Then, the network learns to map the low resolution input to the higher resolution original image.

A recent example is by Ledig et al., (2017), who utilize Generative Adversarial Networks (Goodfellow, Pouget-Abadie, et al., 2014). Concurrently, a *generator* network and a *discriminator* network are trained. Generator learns super-resolution and discriminator learns whether the input is an original or a super-resolution image.

Lai et al., (2017) utilize transposed convolution to increase the resolution of both the image and its features, which are fused together by summation.

Dong et al., (2016) extract features from low resolution input, non-linearly maps to a higher resolution then reconstructs the features, obtaining super-resolution output. Another idea is by Behnke, (2001); a convolutional layer outputs 4 channels from an input image, which are rearranged into an output with twice the resolution of the input. This rearrangement can be interpreted as a transposed convolution with fixed weights.

A 3D example can be given by Pham et al., (2017), who increase the resolution of brain MRI images by first upsampling and performing series of 3D convolutions on the upsampled input.

These methods involve either serialization of output, transposed convolutions

or upsampling. While the outputs of these methods may look unnatural due to distorted colors, most are successful at sharpening the edges which is suitable for our task.

4.3 Class Imbalance

Roots are distributed sparsely inside the soil. Thus, the proportion of the number of non-root voxels to the number of root voxels in the ground truth is extremely large. Usually, when such class imbalance is not handled, it causes the network to ignore the minority class (root) as noise and be biased towards majority class (non-root). There are several approaches for this problem. Usually, this is either on the basis of the loss function or altering the distribution of the training samples.

Tahir et al., (2009) uses *undersampling* to eliminate some of the samples from the majority class. This method can solve the problem, however, it also causes the training dataset to shrink.

Instead of undersampling the majority class, the minority class can be oversampled. SMOTE (Chawla et al., 2002) artificially increases the number of minority class samples by adding random noise to its features.

Thai-Nghe et al., (2010) propose *cost sensitive learning* which weighs the loss of majority and minority classes differently. A con of this method could be the selection of proper parameters for training.

Another way is to use a loss function that deals with class imbalance directly. Intersection over Union (IoU) is a metric robust against class imbalance. Due to the necessity of thresholding, it cannot be directly used to train networks. Rahman and Y. Wang, (2016) introduce IoU loss which deals with the class imbalance by training on the approximation of the intersection over union. This is explained with more details in section 6.4.1.

4.4 Root Detection

One of the goals of this project is to help the automatic extraction of the structural model of the root. Schulz et al., (2012) developed such a structure extraction algorithm. First the tubular structures are detected in the root, then the root connectivity is determined by treating the root structure as a tree. After the extraction of root structure, the properties of the root such as local diameter and mass are determined. The algorithm works better with less noisy and higher resolution images. By segmenting the MRI images in super-resolution, it is expected for the algorithm to perform better.

A modified version of Schulz et al., (2012)'s algorithm have been implemented in the NMRooting (Dusschoten et al., 2016) software. We test our segmentations with this software for automated root detection.

Using 3D CNNs, Horn, (2018) developed architectures for super-resolution segmentation of the plant root MRI images. This involves the use of upsampling operations for super-resolution. It has been observed that while 3D CNNs can achieve relatively accurate results, training them is difficult as they are highly volatile to different parameterizations such as learning rate, kernel size, number of channels, and number of layers. Moreover, the memory requirements are high and training is slow.

5 Real & Augmented Data

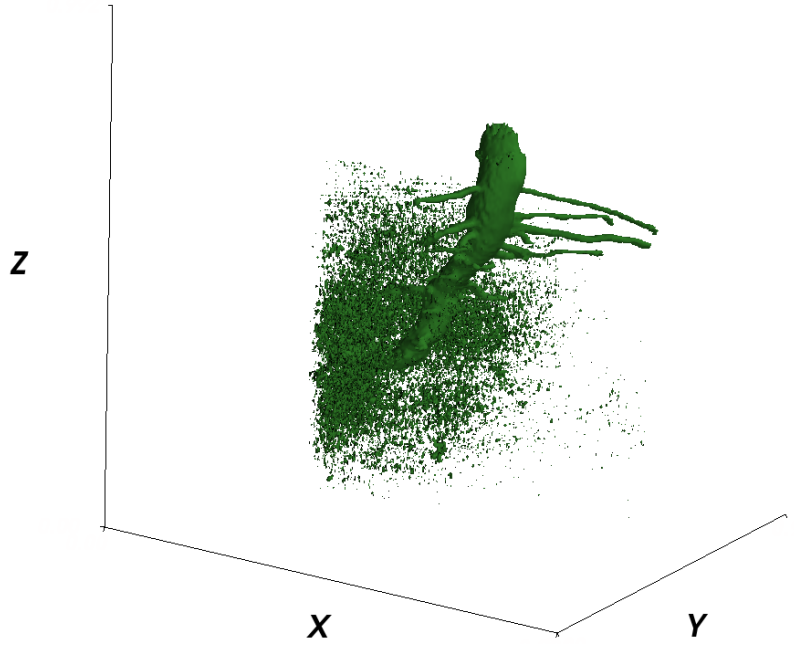


Figure 5.1: Thresholded visualization of the *Lupine Small* MRI scan.

The dataset is a major issue for this project. We want to train the network with supervised learning. This requires end-to-end mapping of low resolution MRIs to high resolution segmentation ground truths. Unfortunately, during the time of this thesis, such data is not available.

The data we are given includes 4 pairs of plant root MRI (see table 5.1) and for each of them, an XML file containing structure of the root. Apart from this file, no usable teacher data are available.

Dataset Name	Resolution on axes			Size of axis (mm)			Usable
	x	y	z	x	y	z	
Lupine Small	256	256	128	40	40	40	Yes
Lupine 22 August	256	256	120	100	100	129	Yes
Lupine April 2015	256	256	128	64	64	70	Partly
GTK	183	183	613	28	28	100	No

Table 5.1: The plant root MRI scans that we are given

5.1 MRI Files

Our original data contains 4 MRI scans, of which only 2 are usable (table 5.1). The other 2 files are unusable as they are corrupted. Each voxel contains an intensity value in range [0, 1]. The format of these files is VTK (Schroeder et al., 2004); which allows visualization of the volume under various software such as ParaView (Ahrens et al., 2005).

3 of them are inside a pot and all 4 have a test tube inserted into the soil. In figure 5.1, ISOSurface with 50% intensity threshold of the MRI *Lupine Small* is displayed. While much of the noise in soil is still visible, due to thresholding, the roots either appear thinner than reality or disappear completely.

The details of these four MRI images are given in table 5.1. In figure 5.1, the bottom and top of the plant root lies on the extremes of the z axis. Similarly, throughout this thesis, z axis refers to the axis that follows the scan from the top to bottom.

Previously, we intentionally referred to the voxels of the MRI scans as cuboids instead of cubes. This is due to differing unit length of the voxels on different axes. For example, the x and y axes of Lupine Small voxels are (40/256)mm; while on the z axis this is (40/128)mm.

5.2 Data Augmentation

Originally, there are 2 usable real MRI images. The ground truth for these images can be generated by voxellizing the structure denoted in the XML files. However, this brings a couple of issues. First, as described in section 5.2.1, the ground truth and the real MRI images do not match perfectly due to the misalignments found in the ground truth. In the ground truth, if the roots are misaligned or not annotated at all, the learning algorithm can learn the wrong goal. The second issue is insufficiency of the data. Training a neural network in a supervised manner

requires large amount of labelled data. 2 pairs of training samples is not enough to train large networks. Moreover, the network needs to be tested with real MRI images that are not part of the training set. Thus, we opt for synthesizing our own data for training and validation. We reserve the real MRI scans and their ground truths as the test set.

Synthetic plant root MRI is generated by introducing variety to the dataset through 3 rotations $rot \in \{0°, 60°, 120°\}$, mirroring on 2 axes (x and y), x-y axis swapping and modifying the root thickness by multiplying the root thickness with $r_f \in \{0.34, 0.71, 1, 1.41\}$. Further variety is provided by generating soil noise modeled after real MRIs. 2 noise types are modelled after Lupine Small and Lupine 22 August under 5 different intensity scales so that the network can adapt to different SNRs. Using four XMLs, this provides us with $4 \cdot 4 \cdot 3 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 5 = 3840$ input MRI, ground truth pairs.

The overall procedure of dataset generation is given in algorithm 1.

5.2.1 Root Structure XML Files

Each MRI file listed in table 5.1 has a corresponding XML file. This XML file describes a tree structure where each node of the tree contains a position in 3D space and the radius of the root at this specific position. We define root branch as a part of the root which starts from a leaf node (branch tip) and ends at the root node (plant shoot).

These XML files can be used to voxellize root structures. However, as these files are generated under human supervision, the location and thickness of the branches are often incorrect. Some branches are missing altogether. Additionally, the position of roots are given relative to the position of the plant shoot. Yet, the XML files provide no information regarding the shoot positions. This necessitates manual alignment of the plant shoot manually to the shoot of real MRI scans.

Regarding the branch thicknesses, the XML files only include radius. This makes estimation of non-circular shapes impossible.

5.2.2 Generation of Occupancy and Intensity Grids

Using the XML files (see section 5.2.1), each root branch is fitted with a circular tube following the positions defined in its nodes. This can be interpreted as sweeping a 3D volume from the branch tip to the plant shoot using a circle whose diameter is adjusted with the radius given in the nodes. As the nodes lie discretely in 3D space, the positions and the radii between two nodes are estimated with cubic spline interpolation and linear interpolation respectively.

The voxelization of this structure is not straight-forward as there exists no known (to us) analytical way of exact calculation of the occupied part of a voxel by this tube. Therefore, another method is used to closely approximate the occupancy of a voxel. This is done by creating a grid of points (e.g. $4 \times 4 \times 4$) inside the voxel and checking the fraction of points which lie inside the root structure.

Given an MRI image of resolution (x, y, z), we initialize an occupancy grid $V_{ultra} \in \{0, 1\}^{8x \times 8y \times 8z}$, and an intensity grid $I_{ultra} \in [0, 1]^{8x \times 8y \times 8z}$ with 0s. For each root branch, a random number $rnd \sim U([0.5, 1])$ is sampled, for each voxel v at index (i, j, k), if the 3D position v_{pos} lies inside the fitted tube, then,

$$V_{ultra}^{i,j,k} = 1 \tag{5.1}$$

$$I_{ultra}^{i,j,k} = rnd. \tag{5.2}$$

To obtain a super-resolution occupation grid and a normal resolution intensity grid, downsampling by the factors of 8 and 4 are applied to V_{ultra} and I_{ultra}, respectively. The downsampling function (algorithm 1, lines 8 and 10) can be given as:

$$B = downsample\_by\_n(A) \tag{5.3}$$

$$B^{i,j,k} = \frac{1}{n^3} \sum_{i',j',k'=0}^{n-1} A^{i*n+i',j*n+j',k*n+k'} \tag{5.4}$$

In the augmented MRI images, each voxel is represented by $8^3 = 512$ points in I_{ultra} and each voxel in the super resolution ground truth $G \in [0, 1]^{2x \times 2y \times 2z}$ is represented by $4^3 = 64$ points in V_{ultra}. The super-resolution ground truth is obtained by thresholding the super-resolution occupancy grid. I.e., if half of a voxel is occupied, then it is marked as root in the ground truth. I.e.,

$$G^{i,j,k} \leftarrow \begin{cases} 1 & \text{if } V^{i,j,k} \geq 0.5 \\ 0 & \text{otherwise} \end{cases}. \tag{5.5}$$

5.2.3 Noisy Image Generation

It is desirable to generate augmented MRI root images that are as close to real data as possible. As described in table 5.1, only *Lupine Small* and *Lupine 22 August* are completely usable. Thus, we model our noise distributions after these 2 MRI images.

Figure 5.2: An example occupation grid for Lupine 22 August with $r_f = 0.34$. The visualization is 0.5 intensity ISOSurface.

Lupine Small Various connected noise structures are observed which are described in the figure 5.4. In addition to the noise, the plant is inside a pot and has a vertically inserted test tube. Outside the pot, we do not notice any connected structures. However, an MRI artifact that blurs the whole image can be seen.

For implementation, we make use of Perlin noise (Perlin, 1985). Perlin noise is a gradient-based method that is used extensively in the field of computer graphics. It can be used to generate random terrain, texture or smoke-like structures in any number of dimensions; which makes it useful for our task. Usually, this involves sampling in different scales and fusing them by summing together. An example can be seen in figure 5.3.

In this case, the Perlin noise is used to create both large high intensity areas and small blob like structures. The procedure is very complicated, thus, it is given with its parameters in the algorithm 2.

In line 1, the tensors C and H are initialized with zeros. Tensor C is used to generate large random, irregularly shaped blobs while H is used to generate small irregularly shaped blobs. In lines 2 to 5, both C and H are initialized with same Perlin Noise distribution. For generation of large blobs, we use Perlin noise only

as a non-binary mask. To this end, its standard deviation and mean are adjusted and clipped between 0 and 1. At this point, the noise structure C contains large blobs with inner areas equal to 1, and borders between 0 and 1. In line 8, we further increase the contrast of the borders so that the passages between large intensity and low intensity areas are stark. We observe from the real MRI scans that the inner areas of these large areas contain a distribution similar to a normal distribution. Thus, each voxel of C is multiplied by a random normal sampled value (line 9). From lines 10 to 13, 20000 Gaussian blobs are generated. Each Gaussian blob has a randomly sampled location and scale parameter. It is desired that these Gaussian blobs not only increase the intensity but also decrease, thus, their scaling value also includes negative numbers. The small blobs that are found in the real MRI scans have high frequency intensity, to grant this, the intensities of each voxel of H are adjusted in lines 14 to 16. The empty areas of the real MRI scans contain noise that is similar to uniform distribution, further voxel-wise noise is added to H to have this effect. Finally, in line 19, the small blob and large blob structures are fused together by summation, followed by clipping which ensures values are in range [0,1].

Lupine 22 August The noise of Lupine 22 August does not seem to have connected structures with the exception of some MRI artifact. Much of the noise has a truncated normal distribution without any noticeable connected structures. Similar to Lupine Small, it contains a bent test tube inside. The procedure is given in algorithm 3.

Combination with the intensity volume After generation from a noise type, it is multiplied by a scaling factor $s \in [0.2, 0.4, 0.6, 0.8, 1.]$. This is done to force the learning algorithm to adapt to different SNR levels. The intensity volume and the scaled noise is fused together by summation followed by a voxel-wise intensity clipping, ensuring each voxel intensity lies in range [0, 1].

With Lupine Small, we also notice an MRI artifact which blurs the image across the x axis. Upon the combination of the noise with the intensity grid, we apply blurring with a $1 \times 5 \times 1$ kernel to simulate this effect.

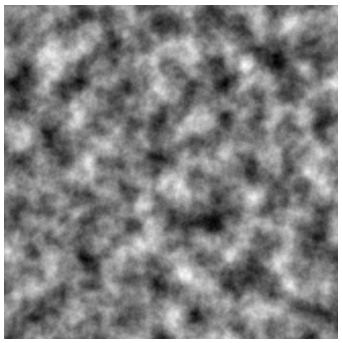


Figure 5.3: A 2D Perlin noise example

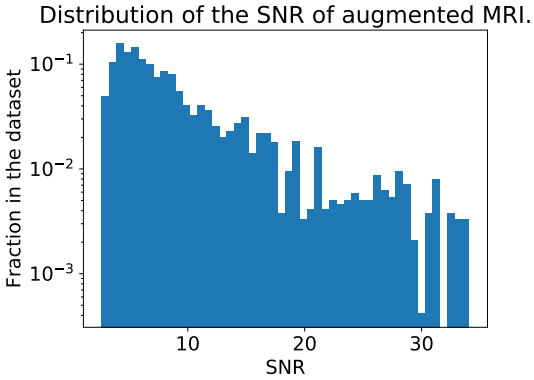


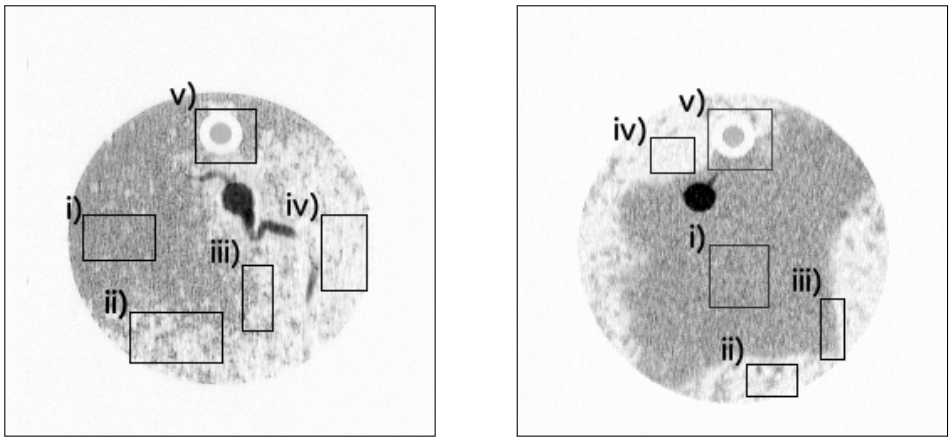
Figure 5.7: Histogram of the Dataset SNR. Lower is noisier.

5.3 SNR

Signal-to-noise ratio (SNR) is defined as the ratio of the power of the desired signal (root) to the power of the background noise. This power ratio can be defined with different formulations. For an MRI image I , let $mean(I_r)$ be the mean value of root voxel intensities and $std(I_s)$ be the standard deviation of non-root voxels of I. Then, SNR of I can be defined as:

$$SNR = \frac{mean(I_r)}{std(I_s)} \tag{5.6}$$

as given by Higgins, (2003-2018). SNR depends on the underlying noise distribution, random intensity scaling and thickness of the branches. Upon investigation of our augmented data, we observe that the lower SNR corresponds to thinner roots with high intensity background noise as expected.



(a) A slice of thickness one along z axis from *Lupine Small* real MRI image.

(b) A slice of thickness 1 along z axis from a *Lupine Small* augmented MRI image.

Figure 5.4: The comparison between the *Lupine Small* and its augmentation. The details are as follows:

i) contains large high intensity areas. These are the result of a Perlin noise used a mask for a uniform noise.

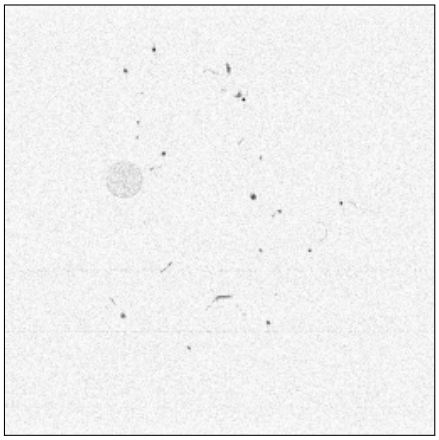
ii) contains blob-like 3D connected structures. These are synthesized by applying various thresholding operations on Perlin noise.

iii) contains areas with high spatial contrast.

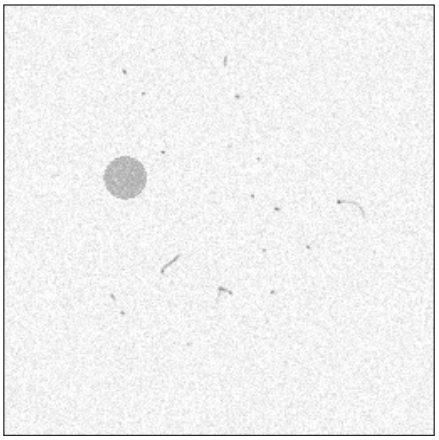
iv) marked areas are the result of low intensity uniform distribution.

v) contains a test tube with two parts. For each voxel a random value from a truncated normal distribution is added.

- The pot follows an almost circular shape along z axis; a cylinder that is perpendicular to an x-y plane is positioned as a mask to the noisy image. The areas outside the cylinder get low intensity values, sampled from a truncated normal distribution.



(a) A slice of thickness one along z axis from *Lupine 22 August* real MRI.



(b) A slice of thickness one along z axis from *Lupine 22 August* augmented MRI.

Figure 5.5: The comparison between the *Lupine 22 August* and its augmentation. *Lupine 22 August* does not have a pot. The test tube is still present. A truncated normal distribution is used to set the intensity of the test tube.

Algorithm 1: The generation of the dataset; main procedure.

Data: XML root structure x, boundary
$min_x, min_y, min_z, max_x, may_y, max_z \in \mathbb{R}$, resolution $r \in \mathbb{Z}^3$
Result: Set of augmented MRI - super-resolution ground truth pairs

$D \leftarrow \emptyset$
$N \leftarrow read\_all\_nodes(x)$
foreach $r_f \in \{0.34, 0.71, 1, 1.41\}$, $rot \in \{0°, 60°, 120°\}$ **do**
  $N' \leftarrow \emptyset$
  foreach *node* $n \in N$ **do**
    $n' \leftarrow rotate\_around\_center(n, rot)$
    $n'_{radius} \leftarrow n'_{radius} * r_f$
    $N' \leftarrow N' \cup n'$
  `/* Generate very high dimension grid, check every point if they are occupied` V`. ` I_{super} `is a high resolution intensity which provides randomness to the intensity to different branches. See section 5.2.2. */`
  $V_{ultra}, I_{ultra} \leftarrow generate\_volume\_ultra\_resolution(N', b_x, b_y, b_z, r)$
  $V_{super} \leftarrow downsample\_by\_4(V_{ultra})$
  $G_{super} \leftarrow threshold\_0.5(V)$
  $I \leftarrow downsample\_by\_8(I_{ultra})$
  foreach $x_m, y_m, swap\_axes \in \{0, 1\}$ **do**
    $I' \leftarrow I, G' \leftarrow G_{super}$
    if $x_m = 1$ `// mirror the images on x axis`
     then $I' \leftarrow mirror\_x(I'), G' \leftarrow mirror\_x(G')$;
    if $y_m = 1$`// mirror the images on y axis`
     then $I' \leftarrow mirror\_y(I'), G' \leftarrow mirror\_y(G')$;
    if $swap\_axes = 1$`// Swap the x and y axes`
     then $I' \leftarrow swap\_axes(I'), G' \leftarrow swap\_axes(G')$;
    foreach $noise\_type \in \{lupine\_small, lupine\_22\}$ **do**
      foreach $noise\_instensity \in [0.2, 0.4, 0.6, 0.8, 1]$ **do**
        $I_{noisy} \leftarrow generate\_noise(I, noise\_type, noise\_instensity)$
        $D \leftarrow D \cup (I_{noisy}, G')$
return D

Algorithm 2: The generation of the noise modelled after Lupine Small.

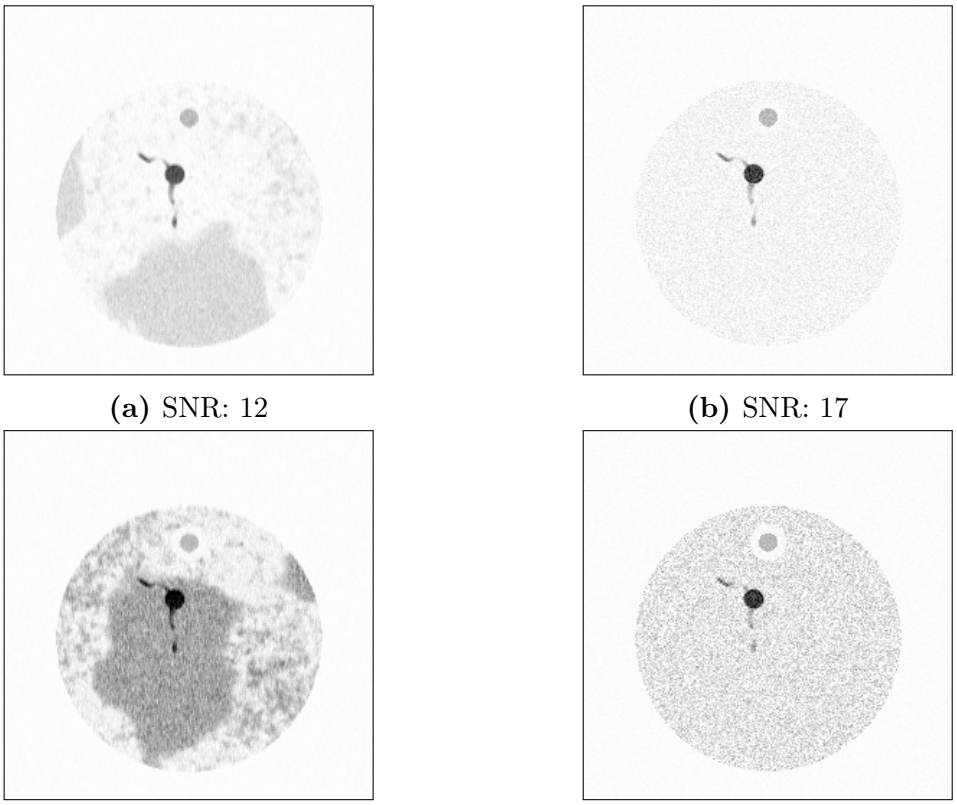
Data: Resolution $r \in \mathbb{Z}^3$
Result: Set of augmented MRI image - super-resolution ground truth pairs

$C, H \leftarrow zero\_filled\_tensor\_of\_shape(r)$
foreach $i \in [1, 6] : i \in \mathbb{Z}$ **do**
  $scale = r * 0.5^i$
  $C \leftarrow C + random\_perlin\_noise(r, scale)/2$
  $H \leftarrow H + random\_perlin\_noise(r, scale)/2$
$C \leftarrow C \cdot 1.6/std(C) - 0.6$
$C \leftarrow clip(C, 0, 1)$
$C \leftarrow 1 - (1 - C)^3$
$C \leftarrow C \cdot iid\_normal\_distribution(shape : r, mean : 0.35, scale : 0.15)$
`/* Number of gaussian blobs */`
$count \leftarrow 20000$
`/* Range of possible center locations for gaussian blobs; iid. sampled for each blob */`
$center\_range \leftarrow [(0, 0, 0), r]$
`/* The scale parameter of the gaussian blob; value of the center of the gaussian; iid. sampled for each blob */`
$scale\_range \leftarrow [-0.15, 0.15]$
$H \leftarrow H + random\_gaussian\_blobs(count, center\_range, scale\_range)$
$U \leftarrow random\_iid\_uniform(r, -0.3, 0.3)$
$U \leftarrow U \cdot sqrt((H - min(H))/(max(H) - min(H)))$
$H \leftarrow H + U$
$H \leftarrow clip(H, [-0.02, \infty])$
$H \leftarrow H + random\_iid\_uniform(r, 0, 0.15)$
return $clip(H + C, [0, 1])$

Algorithm 3: The generation of the noise modeled after Lupine 22 August.

Data: Resolution $r \in \mathbb{Z}^3$
Result: Set of augmented MRI image - super-resolution ground truth pairs

$G \leftarrow iid\_normal\_distribution(shape : r, mean : 0.07, scale : 0.07)$
return C **return** $clip(H, 0, 1)$



(a) SNR: 12 **(b)** SNR: 17

(c) SNR: 5 **(d)** SNR: 8

Figure 5.6: The comparison of *Lupine Small* under radius multiplier 0.71 with different noise types and intensities. The first column has the noise modelled after Lupine Small while the second column has the noise modelled after Lupine 22. The first row's noise multiplier is 0.4, and second row's is 1. As expected, the SNR decreases as the noise intensity is increased.

6 Segmentation Method

Recent studies show that, transferring knowledge from a domain to another is useful for semantic segmentation (Yosinski et al., 2014). This is typically called *transfer learning* which involves extracting features from a pretrained network to help learning for another domain.

In this thesis, we utilize 2D networks for segmentation of 3D data. To this end, we employ layer-by-layer segmentation of the MRI root images. A layer refers to a slice of 1 voxel thickness along z axis. For example, the MRI image GTK, whose resolution is $183 \times 183 \times 613$ contains 613 layers along the z axis. For segmentation of the whole MRI image, using a super-resolution factor $k = 2$, each layer is mapped to 2 layers of resolution 366×366. In the end, a final segmentation of resolution $366 \times 366 \times 1226$ is obtained.

As a transfer learning architecture, we take RefineNet (Lin et al., 2017) as our base model. Since RefineNet takes in 2D RGB images for extraction of features from ResNet, a method to map 3D data into 2D RGB images is necessary. This mapping process is more detailed in section 6.3.

6.1 ResNet

ResNet is one of the most successful convolutional neural networks for image classification. It is built on the argument that, as more layers are stacked, the learning should get better. The authors propose a building block (see figure 6.1) for ResNet which sums the input of the block with convolutional layers. This way, in the worst case, the block can act as an identity function, ensuring a lower training error. Due to the use of identity mapping and ReLU activation function, the gradients flow very efficiently through the block, allowing the construction of very deep networks.

6.2 RefineNet

Lin et al., (2017) proposed RefineNet to exploit the information extracted by pretrained networks at different layers of the network. Deep convolutional neural

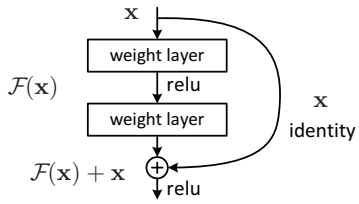


Figure 6.1: A residual block. Image from He et al., (2016)

networks usually follow a series of pooling or strided convolutions, which reduce the dimensionality of the original input image. These layers with small feature maps usually contain complex high level structures, while the earlier layers with larger feature maps contain simpler low level features.

RefineNet makes segmentation by fusing the activations of ResNet using building blocks called RefineNet blocks (figure 6.2). These blocks (see figure 6.2) take

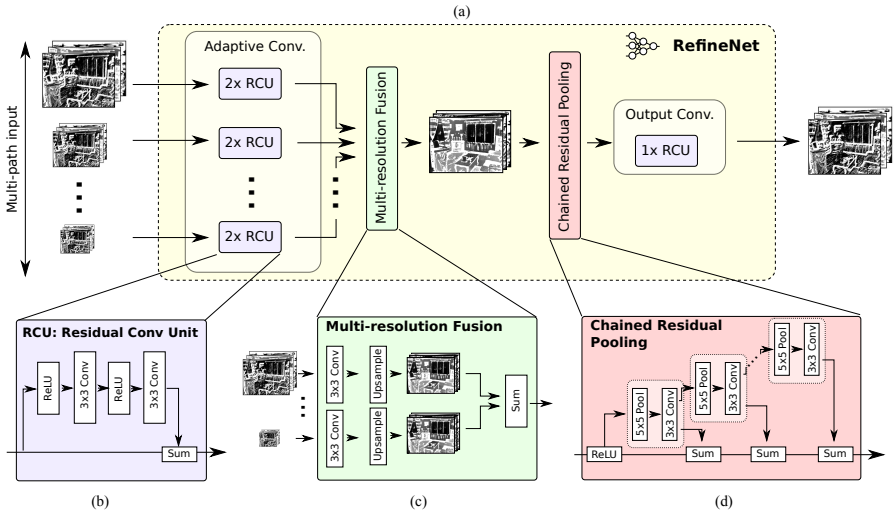


Figure 6.2: RefineNet block. Source: Lin et al., (2017).

in activations of different sizes and generate a high resolution output. A refinement block consists of 3 sub-blocks: Residual Convolution Unit (RCU), Multi-Resolution Fusion, and Chained Residual Pooling. First, Residual Conv Unit (RCU) adapts the activations of the ResNet to RefineNet architecture through a series of convolutions and non-linearities. Multi Resolution Fusion fuses different sized activations by adjusting the number of channels and summing the upsampled convolutions; Chained Residual Pooling applies a series of pooling operations without reducing the resolution followed by a convolution, this enables the capture of larger local area context.

RefineNet architectures are formed with combination of these RefineNet blocks.

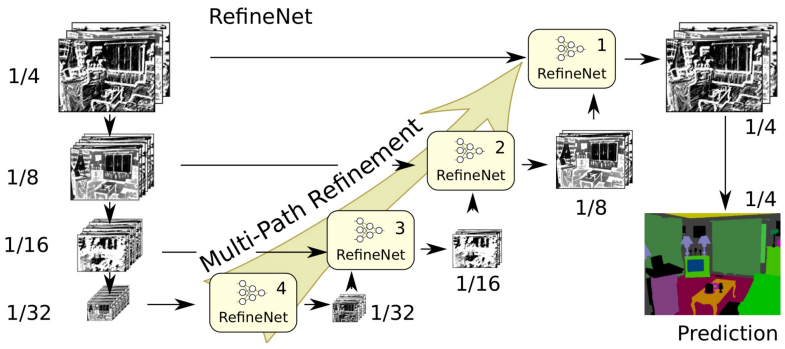


Figure 6.3: 4-Cascade RefineNet architecture. 4 different RefineNet blocks form a cascaded refinement path. Source: Lin et al., (2017).

Lin et al., (2017) test different architectures and find that 4-Cascaded (figure 6.3) architecture gives the best performance.

6.3 Mapping 3D information to 2D

ResNet takes in RGB images as input. Since our data is 3D, it is necessary to map 3D data onto 2D RGB for feature extraction from ResNet.

Let $I \in \mathbb{R}^{x \times y \times z}$ be an MRI image of size and $I^{\cdot,\cdot,l}$ denote lth layer of the MRI among the z axis. Then, for segmentation of layer l, layers in range $\hat{l} \in [l-n, l+n]$, are mapped to $R, G, B \in \mathbb{R}^{x \times y}$ channels.

6.3.1 Averaging

A simple method is the aggregation of layers above l into B, layers below l into R by averaging. The channel G is simply the layer l. This can be formulated as:

$$R^{i,j} = \frac{1}{N} \sum_{k=1}^{n} I^{i,j,l+k} \tag{6.1}$$

$$G^{i,j} = I^{i,j,l} \tag{6.2}$$

$$B^{i,j} = \frac{1}{N} \sum_{k=1}^{n} I^{i,j,l-k} \tag{6.3}$$

By averaging the neighboring layers, it is ensured that information from multiple layers are included. Typically, a neighboring size $n > 2$ is observed to lose details.

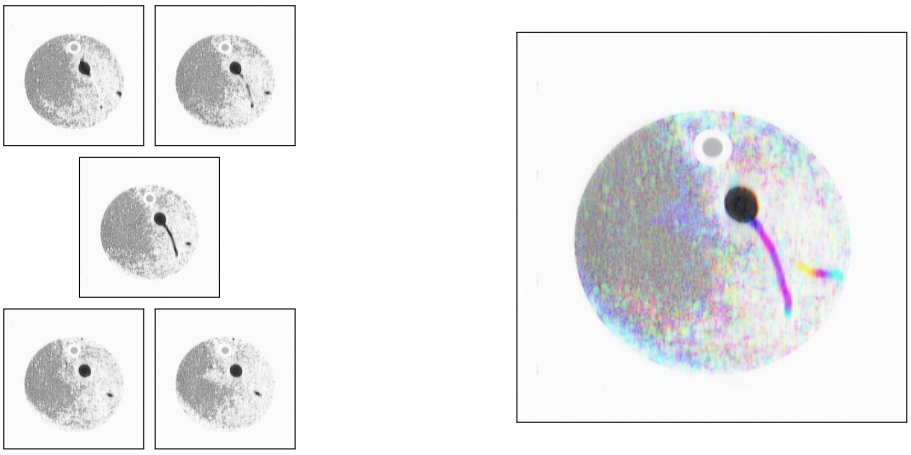


Figure 6.4: Example reduction to RGB, with averaging method where $n = 2$ and $l = 80$. On the left, the layers between 78 and 82 of Lupine Small, and on the right, their reduction to RGB is displayed. All visualizations are negative of their originals.

6.3.2 PCA

PCA is a linear dimension reduction technique. By finding the directions in which the data has the most variance, the larger dimensional data can be represented in a lower dimension with minimal loss.

We apply PCA image-wise. For each voxel v of an image I, a training sample is extracted. The voxel v, n voxels above, and n voxels below - in total $2 \cdot n + 1$ voxels make a training sample.

Or formally:

$$S = \{\{I^{i,j,\hat{k}} \mid \hat{k} \in [k-n, k+n]\} \mid i \in [0,x), j \in [0,y), k \in [0,z)\} \\ : i,j,k,\hat{k} \in \mathbb{N}^0 \tag{6.4}$$

PCA is applied to the set S and projected to 3 dimensions. First, second, and third principal components are mapped to G, R, B respectively. This ordering follows the contribution of each channel to the image luminance.

Example feature maps from ResNet are given in figure 6.5.

6.4 7-Cascade RefineNet

We present a modified version of the original 4-Cascade RefineNet architecture: 7-Cascade RefineNet (see figure 6.6). For feature extraction, we use ResNet-18; a shallower version of ResNet. Due to its shallow structure, ResNet-18 uses much

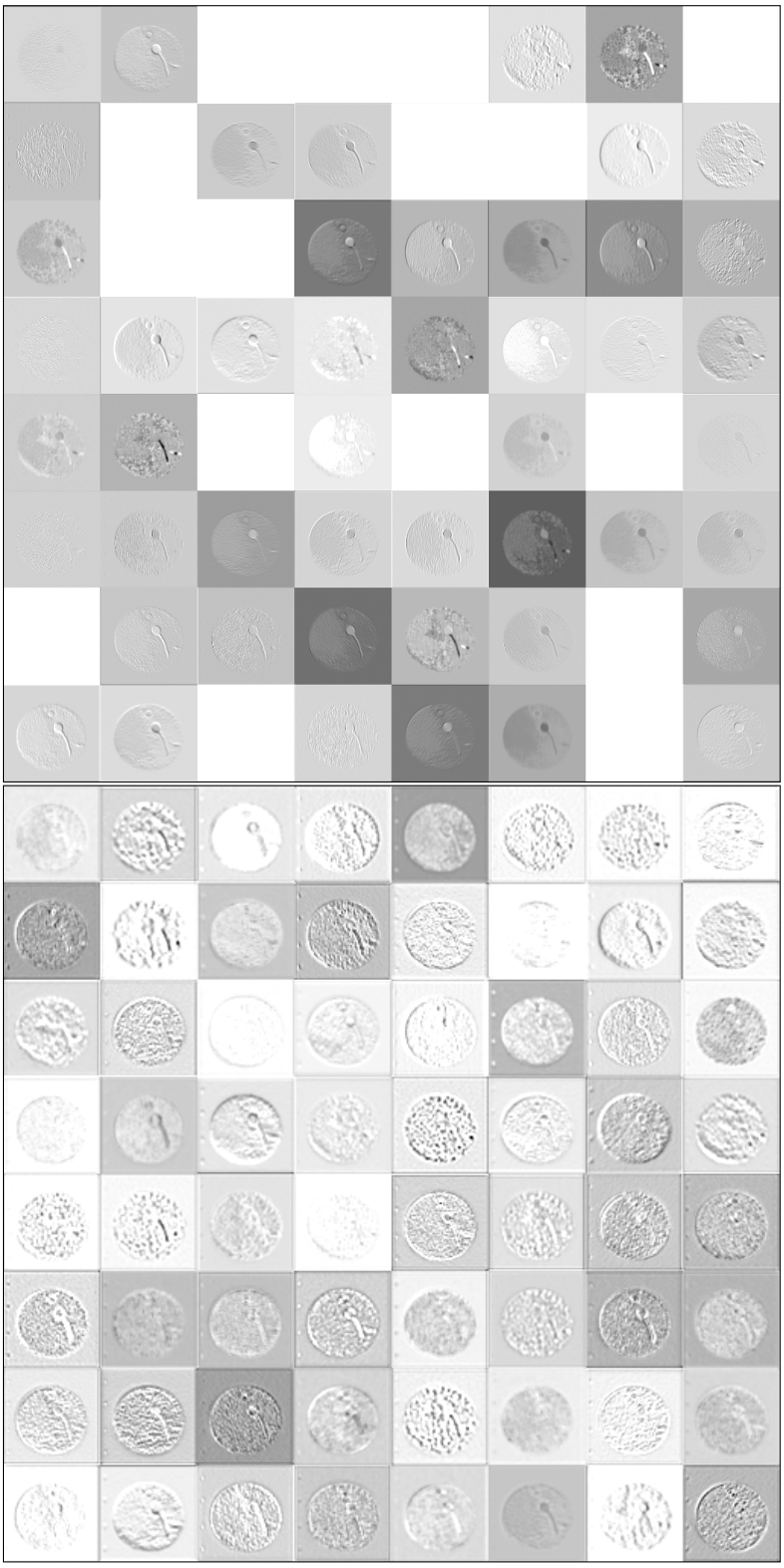


Figure 6.5: Example activations from ResNet-18 using PCA as mapping function. Displayed feature maps belong to 80th layer of Lupine Small MRI image.

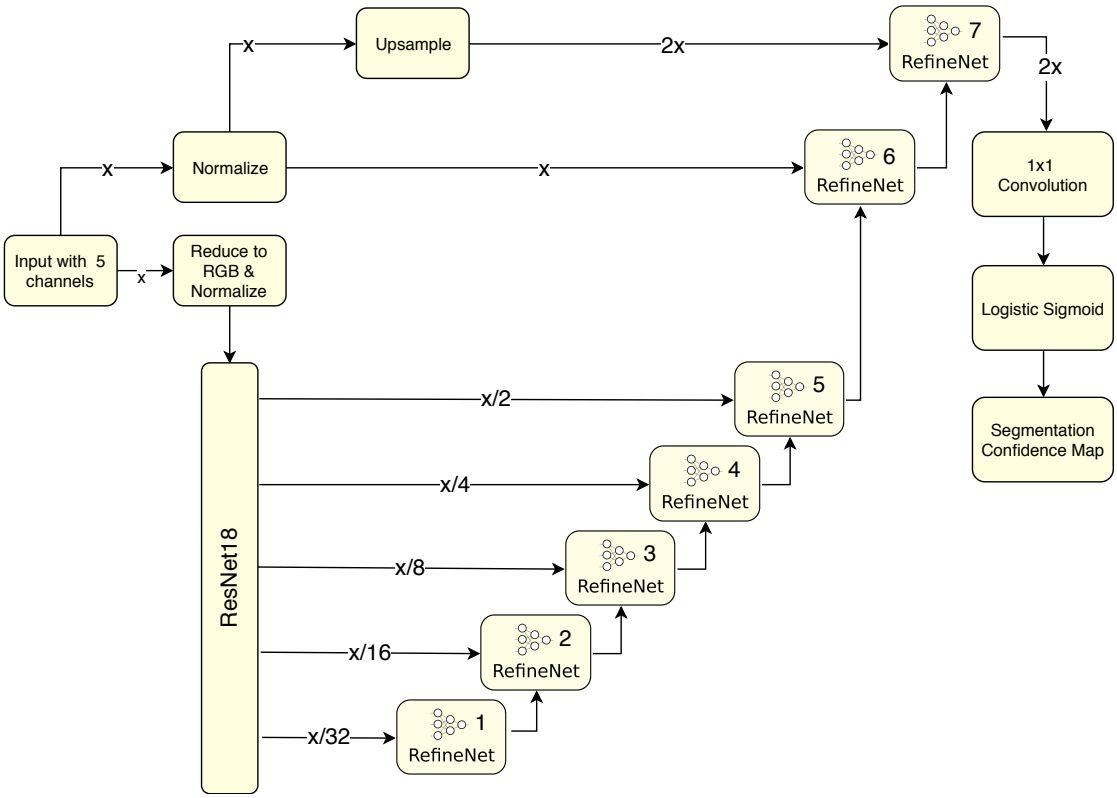


Figure 6.6: Customized 7-Cascade RefineNet Architecture

less memory and runtime is lower. Usually, reducing the network capacity acts as a regularizer and forces the network to learn more general features. This may allow the shallow ResNet-18 to generalize better to the foreign data that we have.

The original RefineNet architecture outputs segmentations that are 1/4 of the resolution of the input while our task is to generate super-resolution outputs. To output higher resolution images, more RefineNet blocks are added to the network. The RefineNet blocks 1, 2, 3, 4 are identical to the ones found in the original 4-Cascade architecture. As higher resolution is needed, we opt for increasing the resolution of the feature maps by a factor of 2 at each RefineNet block. Thus, an extra layer from ResNet whose feature map resolution is half of the original input is fed to the RefineNet block 5. After this point, the RGB feature map is no longer needed as the only reason for reduction to RGB is to extract features from ResNet. Instead, the input whose dimensionality has not been reduced is given as input to the RefineNet 6 block. For segmentation of layer l, with n layers above and below the layer l, each one of the $2n+1$ layers is interpreted as a separate input channel. This way the information that is lost after reduction to the RGB is introduced to the network. The next RefineNet block involves fusing the upsampled input with

the feature maps of the RefineNet block 6, then, a feature map with double the resolution among x and y is obtained. We further look to increase the resolution among the z axis. A final convolutional layer with a kernel of size 1x1 and 2 output channels is introduced. Followed by a sigmoid function (section 3.1.2), these 2 channels are interpreted as 2 consecutive layers of the segmentation where each pixel denotes the confidence of the network that the pixel is a root. For our experiments, we use the threshold 0.5 to decide whether a voxel is root.

With the exception of the RefineNet block 1, all RefineNet blocks output 16 channels each.

For now, this project aims for a super-resolution factor of $k = 2$. To adapt the network for higher super-resolution factors, the upsampling operation before the RefineNet block 7 must upsample by a factor of k and the final 1x1 convolutional layer must output k channels.

6.4.1 Loss Functions

Negative log-likelihood loss

The logarithmic properties of negative log-likelihood overcomes the saturation problem caused by derivatives of the logistic sigmoid function. A naive negative log-likelihood loss for a single classifier $y \in [0, 1]$ and its true class $\hat{y} \in \{0, 1\}$ can be given as:

$$L = -\Big(\hat{y} * log_2(y) + (1 - \hat{y}) * log_2(1 - y)\Big). \tag{6.5}$$

As described in chapter 5, the number of non-root voxels heavily outnumber the number of root voxels and this may cause average root voxel loss to be much higher than average non-root voxel loss. The networks are trained using weighted average per voxel loss; a new variable root weight rw is introduced to adjust the weighting of the roots against non-roots. We define the loss for a mini-batch or a batch as follows:

$$L = \frac{\sum_{i,j,k}(G^{i,j,k} \cdot log_2(Y^{i,j,k}) \cdot rw + (1 - G^{i,j,k}) \cdot log_2(1 - Y^{i,j,k}))}{\sum_{i,j,k}(G^{i,j,k} \cdot rw + (1 - G^{i,j,k}))} \tag{6.6}$$

where G is the ground truth tensor and Y is the tensor corresponding to the confidence values estimated by the network.

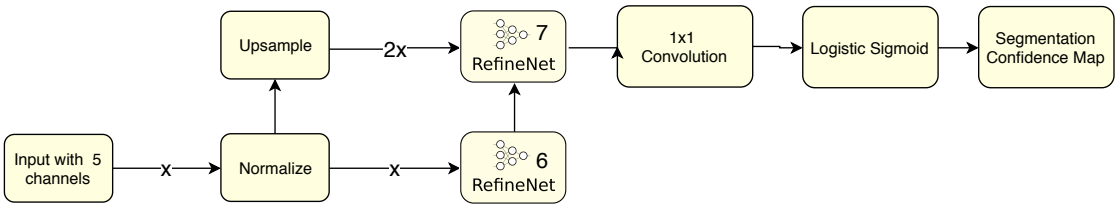


Figure 6.7: RefineNet without transfer learning.

IoU Loss

Apart from the F-Score, intersection over union (IoU) is an alternative metric for imbalanced datasets. Rahman and Y. Wang, (2016) propose an approximation of IoU:

$$\widetilde{IoU} = \frac{Intersection(G, Y)}{Union(G, Y)} = \frac{\sum_{i,j,k}(G^{i,j,k} \cdot Y^{i,j,k})}{\sum_{i,j,k}(G^{i,j,k} + Y^{i,j,k} - (G^{i,j,k} \cdot Y^{i,j,k}))}. \tag{6.7}$$

In terms of a loss function, this can be written as:

$$L = 1 - \widetilde{IoU}. \tag{6.8}$$

The approximation is differentiable, therefore, the loss is also differentiable. One important caveat for such a function is the sigmoid function. When sigmoid function is incorporated into the loss function without its logarithm, it is known to have too small gradients especially when its output is close to 0 or 1 (see figure 3.1), which saturates the learning process.

6.4.2 RefineNet without Transfer Learning

To validate that the network benefits from transfer learning, we remove the pretrained network and the layers, which are fed from the pretrained network (figure 6.7). The rest of the parameters, including the number of channels in the RefineNet blocks (6 & 7 in figure 6.7) are identical.

6.5 Training the Network

6.5.1 Implementation

Due to its dynamic structure and speed, the network is implemented with PyTorch (Paszke et al., 2017). The RefineNet architecture implementation is based on the

code of Fan, (2018). Further modifications have been made on the code to enable the use different sized inputs.

6.5.2 Mini-batches

As displayed in table 5.1, resolution of MRIs are often different. Each layer of the MRI are segmented separately, thus, the resolution of network input is dependent on only x and y axes. For example, $x \times y$ resolution of gtk is 183×183, for other datasets, it is 256×256. MRI layers with different resolutions cannot be processed together in a single batch due to limitations of PyTorch and other deep learning frameworks. Thus, each mini-batch contains 8 layers from the same input MRI.

6.5.3 Gradient Clipping

A common issue when training neural networks is the distribution of the gradients. In our work, it is observed that, at certain times, the loss suddenly increases and converges to some sub-optimal loss and not recover again. Upon investigation of the gradients, we see that these spikes in the loss occur right after a sudden hike in gradient norms.

Our first idea was to use a smaller learning rate, however, using a smaller learning rate, we observed that the loss does not converge to a small value as it did before.

We set a constraint on the norm of the gradients using a method called gradient clipping. If the norm of the gradients is higher than a certain threshold, the gradients are scaled down to fit this norm constraint. Let L be the loss, w be the set of parameters of the network, and t be a certain norm threshold and,

$$g \leftarrow \frac{\partial L}{\partial w} \tag{6.9}$$

then,

$$g \leftarrow \begin{cases} \frac{t \cdot g}{||g||} & \text{if } ||g|| > t \\ g & \text{otherwise} \end{cases}. \tag{6.10}$$

We set the gradient norm threshold to 0.01 which is a rather average value among the gradient norms. We observe that this not only solves the issue of sudden loss spike, it stabilizes the learning process.

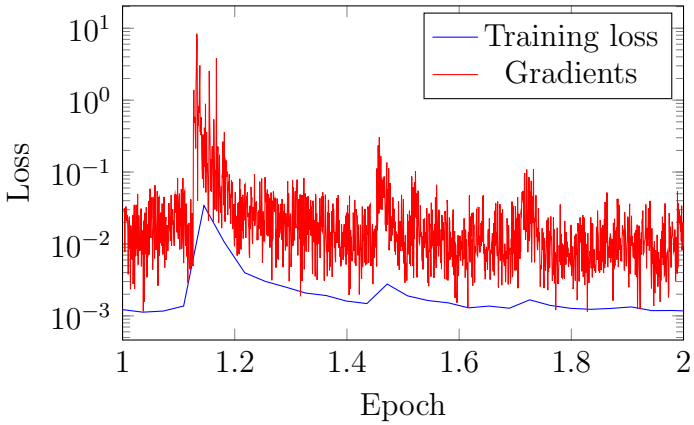


Figure 6.8: The gradients and the training loss between epoch 1 and 2. Upon sudden spike in gradients, the training loss suddenly diverges to a suboptimal value.

6.5.4 Training Algorithm

A mini-batch contains 8 input layer-ground truth pairs from a single MRI image (section 6.5.2). For each training step, average loss of 8 different mini-batches are calculated and backpropagated. In total, a backward pass through the network is processed for 64 input layers. As an optimizer, we use Adam(Kingma and Ba, 2014). By trial and error, we find that 0.0006 is the optimal learning rate for training. Larger learning rates cannot learn at all while smaller learning rates converge to sub-optimal losses.

7 Experiments and Results

We trained several networks to test different variations of input and loss functions. For reduction of volumes to 2D RGB images, we test PCA (section 6.3.2) and averaging (section 6.3.1) methods. Upon comparison (section 7.1), we observe that PCA is the superior method. Moreover, we test the RefineNet architecture without transfer learning (section 6.4.2).

We further investigate the behaviour of the negative log-likelihood (Nll) loss with reweighing and compare with the IoU loss. Since we found that PCA is the superior method (section 7.1), all tests for comparing loss functions have been made using PCA.

Since the the real MRI and corresponding ground truth do not align well with each other, the loss and average F1-Scores are computed on validation set only. The approximate F1-Scores on the test set are presented separately.

In sections 7.1, 7.2, and 7.2.3 different parameterizations of the network are investigated and its results on validation set (augmented) are presented. The test results on real data are discussed in section 7.3.

We decide that the best network is using **PCA** with **negative log-likelihood** loss where root and non-root voxels have **equal weight** $rw = 1$.

7.1 Comparison of Different Input Functions and Effect of Transfer Learning

Our preferred method of mapping 3D data onto 2D RGB is **PCA**. This has been decided by training the network with PCA, averaging method and also without transfer learning. All experiments to compare input functions have been made using negative log-likelihood with $rw = 1$. We observe that PCA is superior to both methods. We believe this is due to PCA containing more information of the 3D data. This also validates the assumption that transfer learning helps. The average F1-Score for network with PCA and $rw = 1$ is **0.95** (see table 7.1).

With averaging method, we do not notice much difference in terms of average F1-Score (**0.93**). For SNRs less than 3.16, PCA achieves 0.89 while averaging method achieves 0.87. For SNRs greater than 3.16 there is no significant difference.

When trained without transfer learning, the average F1-Score is **0.91**. Again, the biggest difference occurs when the SNR is lower than 3.16. The F1-Score drops to 0.84 compared to 0.89 of PCA. This is due to significant drop in recall. Upon qualitative analysis of its segmentations, we observe that even more thin roots are undetected.

Upon qualitative investigation of the MRIs, we observe that the transfer learning helps especially with thin roots as most thin roots are undetected.

7.2 Comparison of Loss Functions

In section 7.1, we see that PCA is the better method of mapping from 3D to 2D RGB data. Thus, when comparing the loss functions, we use PCA as input function.

7.2.1 Training with Equal Weighted Negative Log-likelihood

We train the network using PCA with equal weighted($rw = 1$), loss for root and non-root voxels. The experiments show that the reweighing *may* not be necessary, even with the class imbalance problem. We decide this is the best network for this project.

As expected, the average non-root voxel loss (0.0001918) is proportionally much less than the average root voxel loss (0.0757). These two values are still very small. We theorize, this is due to the noise distribution being very similar across the augmented data. Furthermore, since much of the noise is outside the pot with very low intensities, they can easily be discarded as noise.

The average F1-Score on the validation set is **0.948**. The lowest average F1-Score on the validation set is obtained from Lupine April 2015 (0.911) while the highest is obtained from Lupine Small (0.977). Lupine Small has overall thickest branches while Lupine April 2015 has the thinnest ones. This evidences our prior assumption that thinner branches are harder to detect.

We also investigate validation set with respect to different SNR levels of MRIs. For data with SNR in the range [1, 3.16], the F1-Score is 0.89. For MRIs with SNR above 3.16, the F1-Scores are above 0.97. This may be an acceptable result, moreover, many of the false positives are super-resolution artifacts which can further be ignored.

A closer look at the dataset shows that the common pattern among the worst performed augmented MRI root images is having thin roots with high noise levels. These images are very hard to distinguish even for humans as they may be considered to miss information. Nevertheless, the F-Score, in this case, is still quite

high. We theorize, hard-to-detect nature of these roots may be solving the class imbalance problem by forcing the network to learn roots.

An argument can be made that recall is more important than the precision. The algorithm of Schulz et al., (2012) is successful at eliminating false positives while false negatives are difficult to recover as they are integral to the structure of the root. To this end, in an addition to the F1-Score, F2-Score can also be used.

The loss curve and the accuracy of the network trained with $rw = 1$ and PCA are shown in figures 7.1 and 7.2. Some example segmentations on augmented data are displayed in figure 7.3

7.2.2 Training with Extra Root Weighting

We train the network with $rw = 16$ to trade precision in favor of the recall. While the difference decreases, root loss (0.0160) remains higher than non-root loss (0.00088). As expected, recall increases while precision decreases. Upon qualitative analysis of the segmentations on the augmented data, we observe that the stark reduction in the precision is caused by thicker estimations of the roots. When the roots are very thin, estimating the root even a little thicker than it actually is causes the number of false positives to explode. Thus, lower F1-Score caused by low precision.

The loss curve and the accuracy of the network trained with $rw = 16$ and PCA are shown in figure 7.4. Some example segmentations on augmented data are displayed in figure 7.5.

7.2.3 Training with IoU Loss

We further train the network with a loss function that should automatically deal with the class imbalance problem. An interesting behavior is seen. While the IoU loss is continuously decreasing, during first few epochs, the average negative log-likelihood also decreases. After, the IoU loss keeps decreasing while the average root starts to increase. We theorize this is due to different global minima of IoU and negative log-likelihood loss functions.

The analysis of the F1-Scores on augmented data shows that the IoU loss gives comparable results to training with negative log-likelihood where $rw = 1$. The average F1-Score is 0.943. The resulting loss curve and accuracies are displayed in figure 7.6.

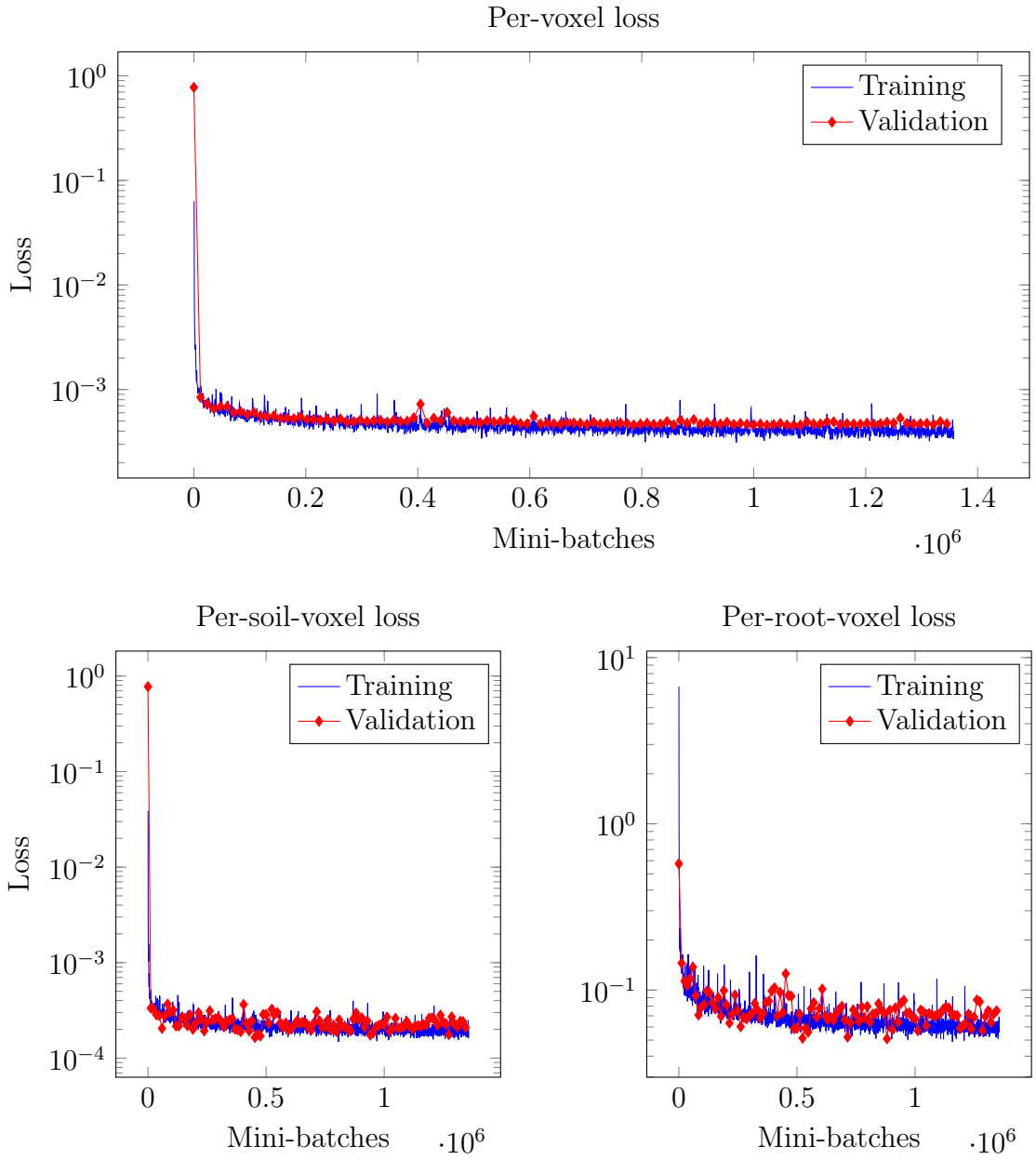


Figure 7.1: Per-voxel loss of RefineNet 7-Cascade trained with $rw = 1$.

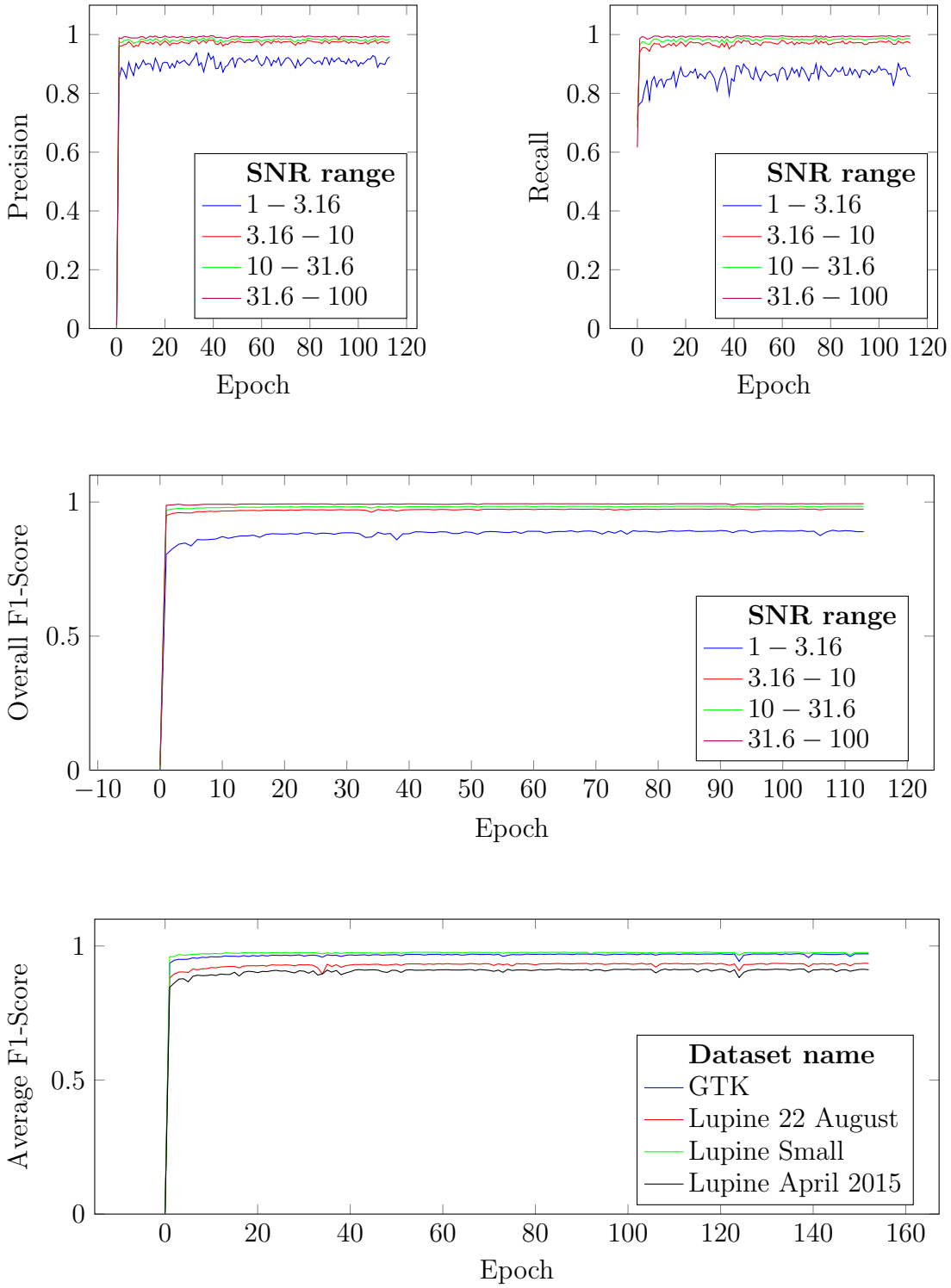


Figure 7.2: Accuracy of 7-Cascade RefineNet trained with negative log likelihood loss where $rw = 1$.

Figure 7.3: Example segmentations of augmented data with very noisy soil environments from 7-Cascade RefineNet, trained using negative log likelihood with $rw = 1$ and PCA as input function. The greens, reds and blues denote true positives, false negatives, and false positives respectively. The segmentations contain almost no false positives with the exception of super-resolution artifacts. However, there are significant number false negatives. Overall root thickness decreases from the first image to the third image. The first image contains only negligible errors. The second image contains minor false negatives in places where the root is very thin. However, the root model extraction software should be able to handle this easily. The third image has significant number of false negatives which cause disconnectivity and even completely missing roots.

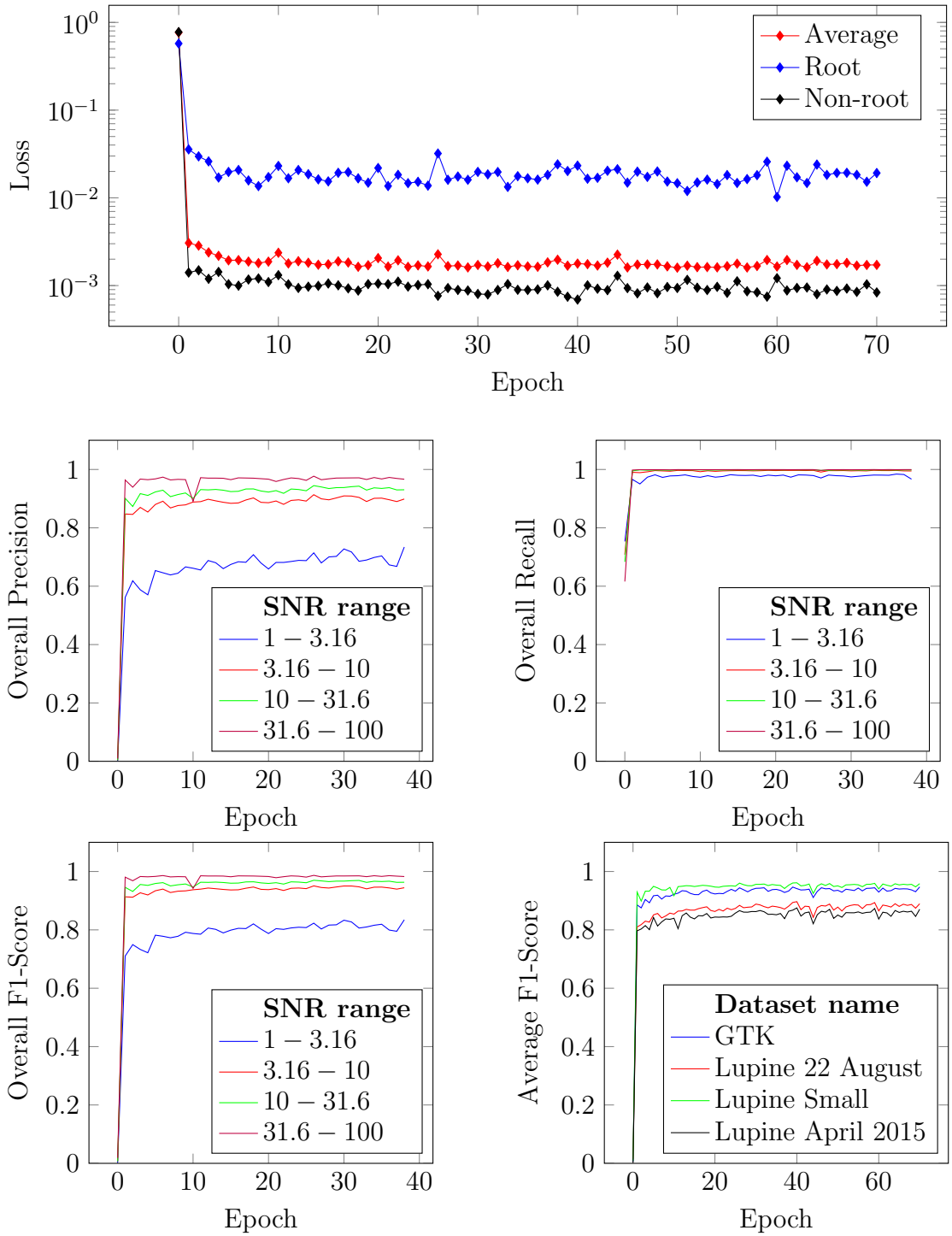


Figure 7.4: Loss curve and F1-scores when the root weight $rw = 16$.

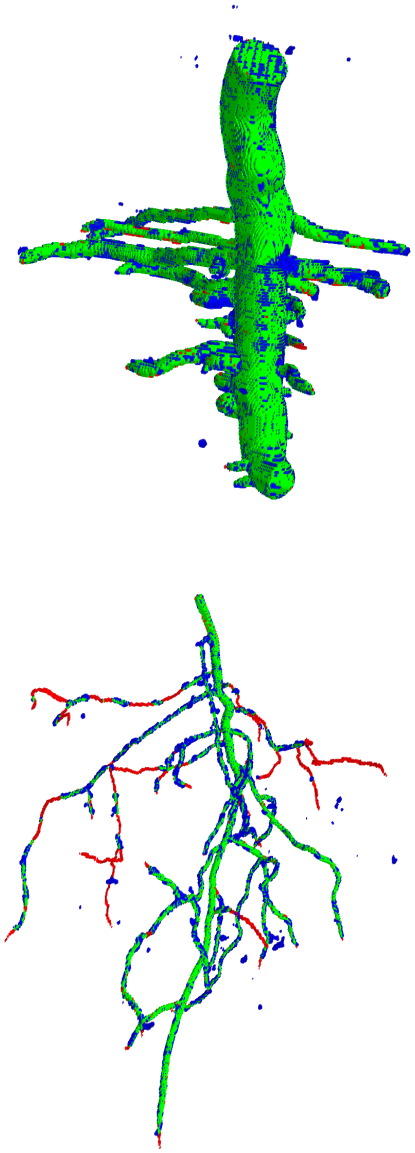


Figure 7.5: Example segmentations from the augmented data with very noisy soil environments. The root weight $rw = 16$. In figure 7.3, there are significant false negatives. After increasing the root weight, more roots are detected. As a downside, the segmentations in the root are thicker than the ground truth. However, these false positives are not as critical as missing roots.
Second image originally has a rather low precision (0.60) since the roots are very thin (see section 7.2.2). With a recall rate of 0.86, the F1-Score is 0.71. We test the same MRI with distance tolerant F1-Score using $dilation = 1$. As expected, the precision increases to 0.91, recall increases to 0.9 giving F1-Score of 0.91. This confirms our expectations that rather low F1-Score is caused by thick estimation of the roots.

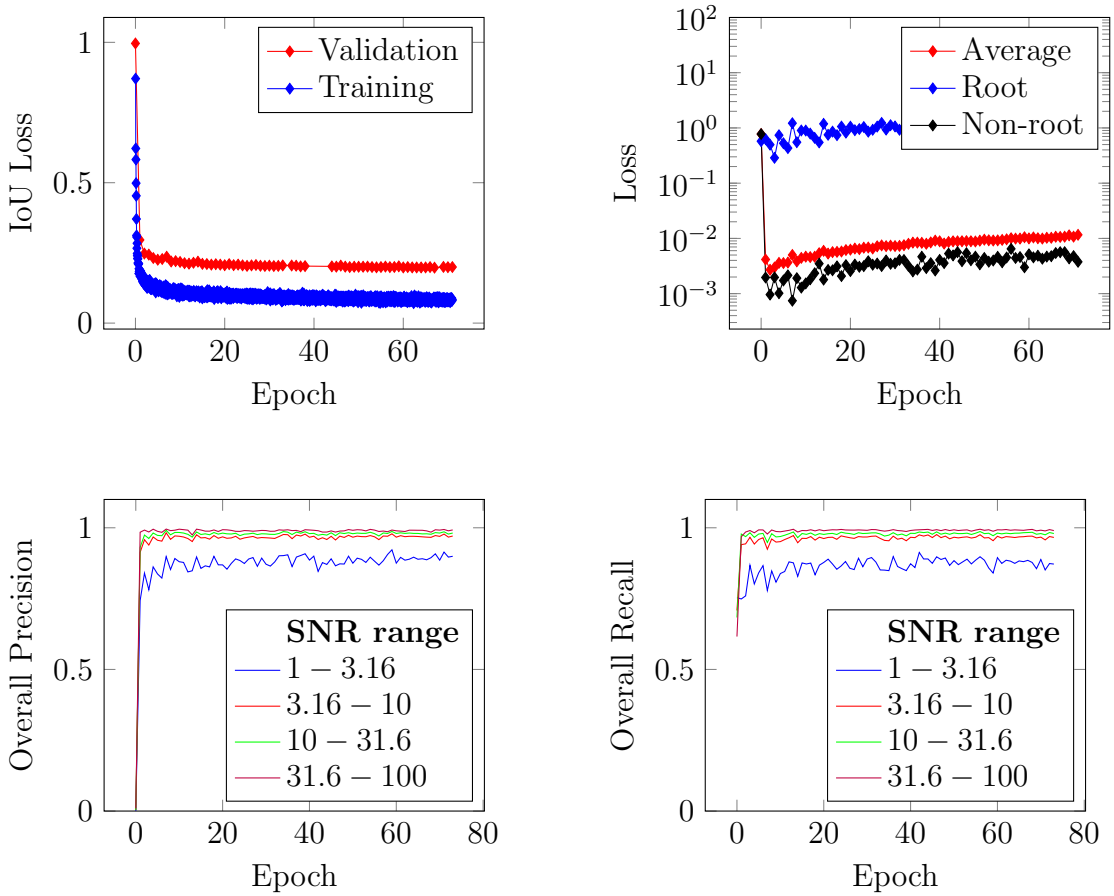


Figure 7.6: Loss curve and accuracy for network trained with IoU loss.

Experiment	Weighted nll loss			IoU Loss	Average F1-Score
	Avg	Root	¬Root		
Nll, $rw = 1$	**0.000457**	0.0757	**0.0001918**	0.218	**0.95**
Nll, $rw = 16$	0.001610	**0.0161**	0.0008790	0.268	0.91
IoU loss	0.009399	1.6274	0.0036990	**0.198**	0.94
Horn, (2018)	-	0.4683	0.004105	-	0.84

Table 7.1: Results on the validation set for different loss functions. The input function is PCA.

Experiment	SNR [1, 3.16]		
	Precision	**Recall**	**F1-Score**
Nll, $rw = 1$	**0.92**	0.86	**0.89**
Nll, $rw = 16$	0.70	**0.98**	0.82
IoU loss	0.89	0.87	0.88

Table 7.2: Results on the validation set for different loss functions. The input function is PCA.

Experiment	Overall F1-Score for SNR range		
	[3.16, 10]	**[10, 31.6]**	**[31.6, 100]**
Nll, $rw = 1$	**0.97**	**0.98**	**0.99**
Nll, $rw = 16$	0.95	0.97	0.98
IoU loss	0.96	**0.98**	**0.99**

Table 7.3: Results on the validation set for different loss functions. The input function is PCA.

7.3 Test on Real Data

We chose the network trained with negative log-likelihood loss using $rw = 1$ and PCA as the input function as our preferred network. We apply our segmentation method to 2 usable real plant root MRI scans that we have: Lupine Small, and Lupine 22 August. Regardless of the loss function, the network gives comparable results. While the trained networks have no difficulty distinguishing thick roots from soil, this is more difficult with thin roots. Due to its thick branches, we are able to obtain precise segmentations from Lupine Small with few false positives. We observe no false negatives, moreover, there are detected root parts which are otherwise not present in the ground truth. These parts are overlooked during human-supervised annotation due to their low intensity. The distance tolerant F1-Scores of Lupine Small can be seen in table 7.4.

In the segmentation of the Lupine 22 August, not many false negatives are present. Unfortunately, the MRI artifacts found in this MRI causes significant increase in the number of false positives. We observe that the segmentation quality changes with respect to the loss function. Negative log-likelihood with $rw = 1$ outputs rather good segmentations, detecting roots that are otherwise not present in the manual annotations. However, there are some minor disconnectivities in the root structure. The distance tolerant F1-Scores of Lupine 22 August can be seen in table 7.4.

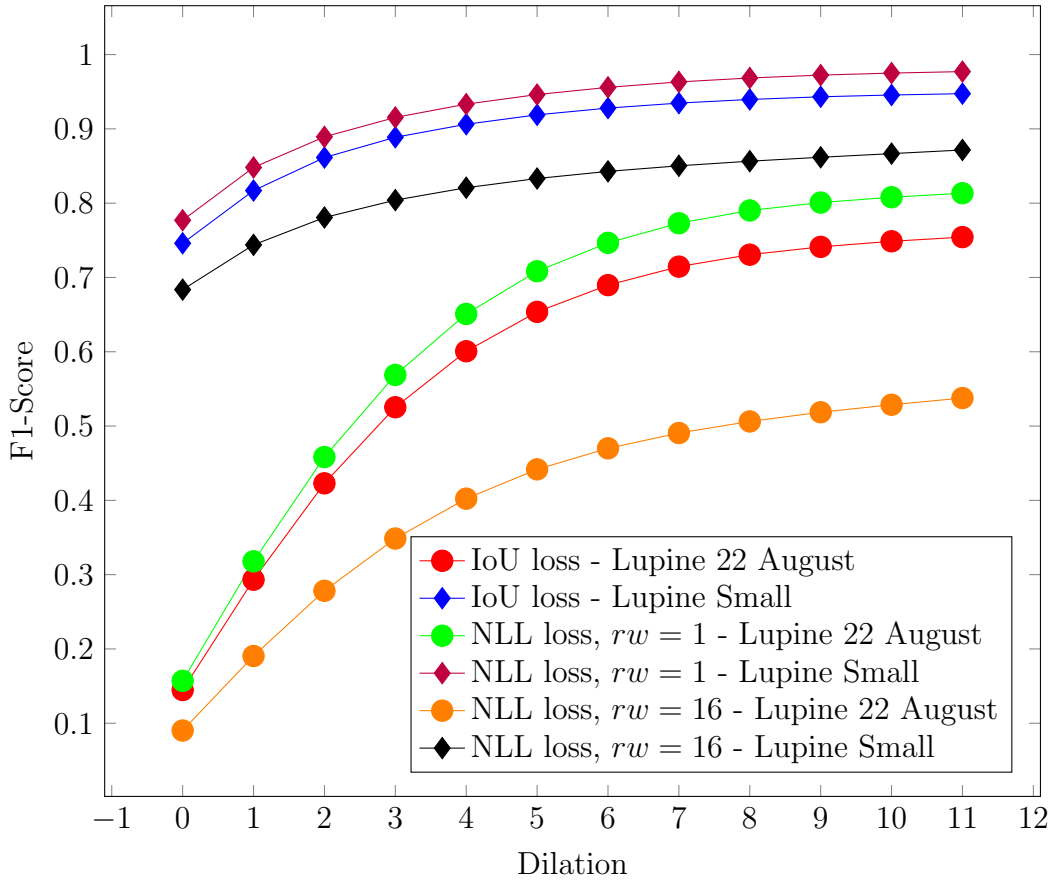


Figure 7.7: Comparison of the networks on real data.

Overall, we can say that our segmentations are better as we have been able to capture extra roots not found in the original manual annotations; are in super-resolution; are more precise.

7.4 Extraction of Root Model

We apply the root model extraction algorithm implemented in NMRooting (Dusschoten et al., 2016). For both Lupine Small and Lupine 22 August, the software successfully extracts the root model and eliminates the majority of false positives while connecting disconnected roots.

Dilation	Precision	Recall	F1-Score	F2-Score
0	0.7891	0.7652	0.7770	0.7699
1	0.8348	0.8617	0.8481	0.8562
2	0.8652	0.9146	0.8893	0.9043
3	0.8875	0.9451	0.9154	0.9330
4	0.9045	0.9639	0.9333	0.9514
5	0.9176	0.9766	0.9462	0.9642
6	0.9278	0.9856	0.9559	0.9735
7	0.9360	0.9919	0.9632	0.9802
8	0.9427	0.9957	0.9685	0.9846
9	0.9480	0.9979	0.9724	0.9875
10	0.9522	0.9990	0.9751	0.9893
11	0.9557	0.9995	0.9771	0.9904

Table 7.4: Quality scores of the network trained with NLL $rw = 1$ on Lupine Small.

Dilation	Precision	Recall	F1-Score	F2-Score
0	0.1164	0.2415	0.1571	0.1987
1	0.2494	0.4383	0.3179	0.3806
2	0.3675	0.6086	0.4583	0.5380
3	0.4617	0.7406	0.5688	0.6608
4	0.5324	0.8371	0.6509	0.7511
5	**0.5834**	**0.9013**	**0.7084**	**0.8128**
6	0.6190	0.9403	0.7466	0.8519
7	0.6450	0.9643	0.7730	0.8774
8	0.6625	0.9787	0.7902	0.8934
9	0.6735	0.9872	0.8008	0.9031
10	0.6816	0.9920	0.8080	0.9092
11	0.6879	0.9947	0.8134	0.9132

Table 7.5: Quality scores of the network trained with NLL $rw = 1$ on Lupine 22 August.

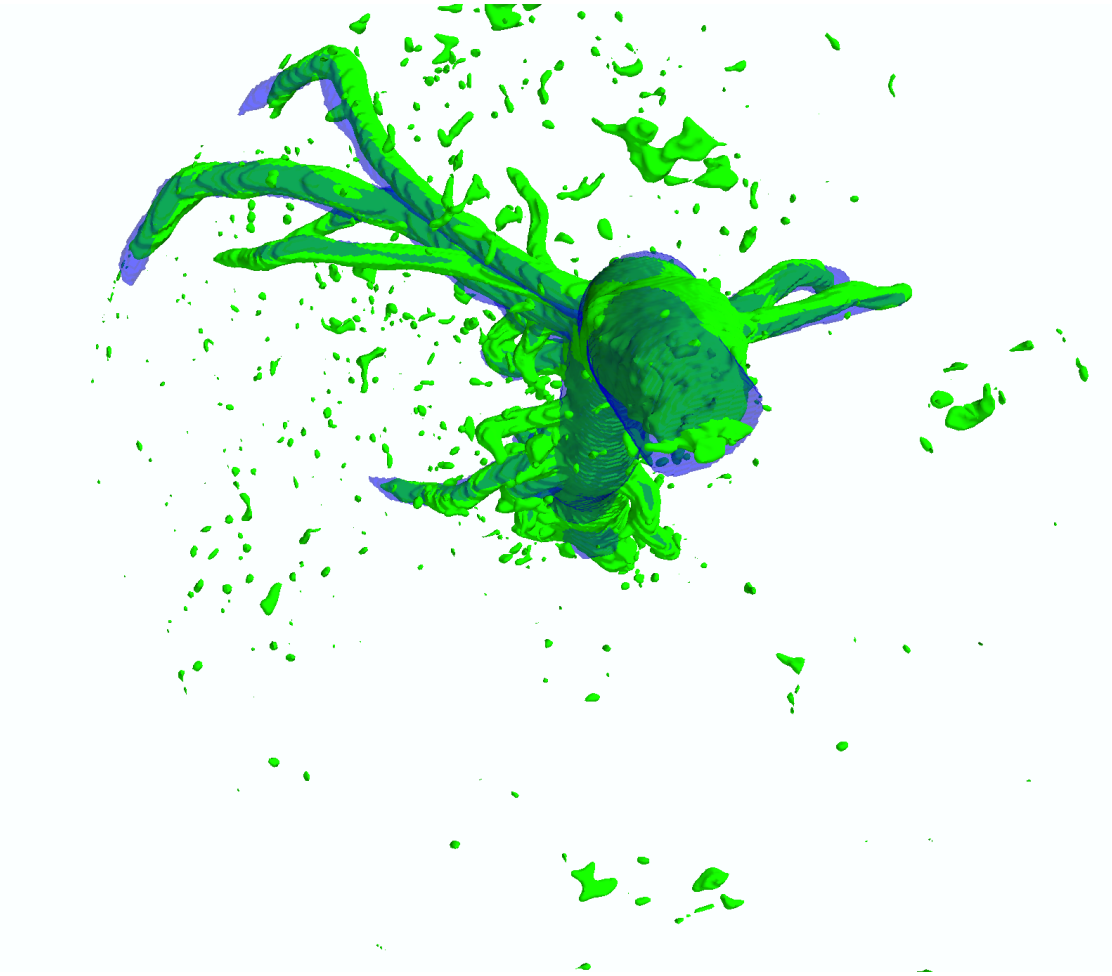


Figure 7.8: Segmentation of Lupine Small. The greens are positive segmentations while the transparent blues are the approximate ground truth. There are very few false positives. Similar to figure 7.9, there are detected root branches which are not present in the ground truth.

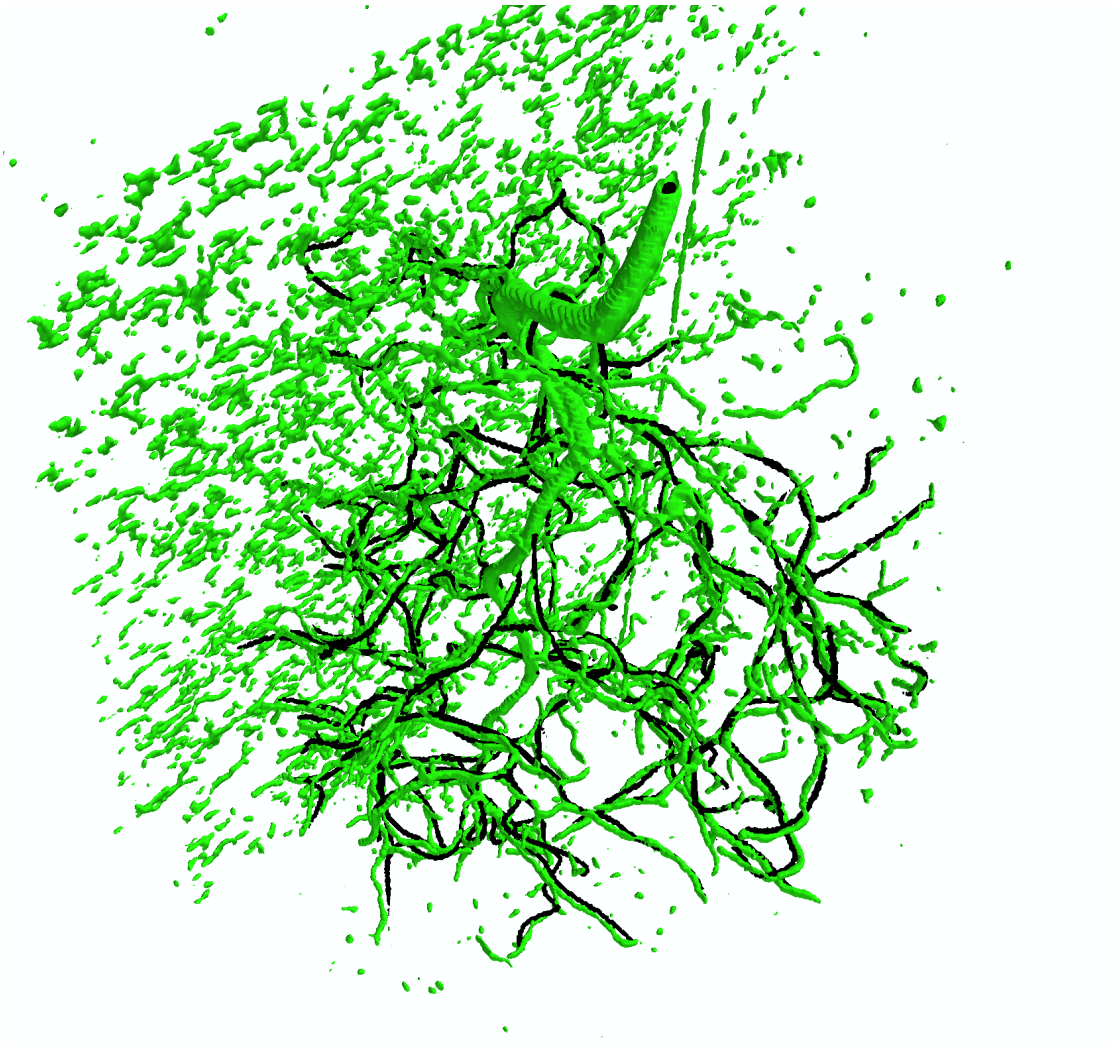


Figure 7.9: Segmentation of Lupine 22 August. The greens are positive segmentations while the blacks are the approximate ground truth. While there are plenty clutters, there are also roots which are not present in the ground truth annotations. As the roots are very thin, even a small misalignment causes low F1-Score.

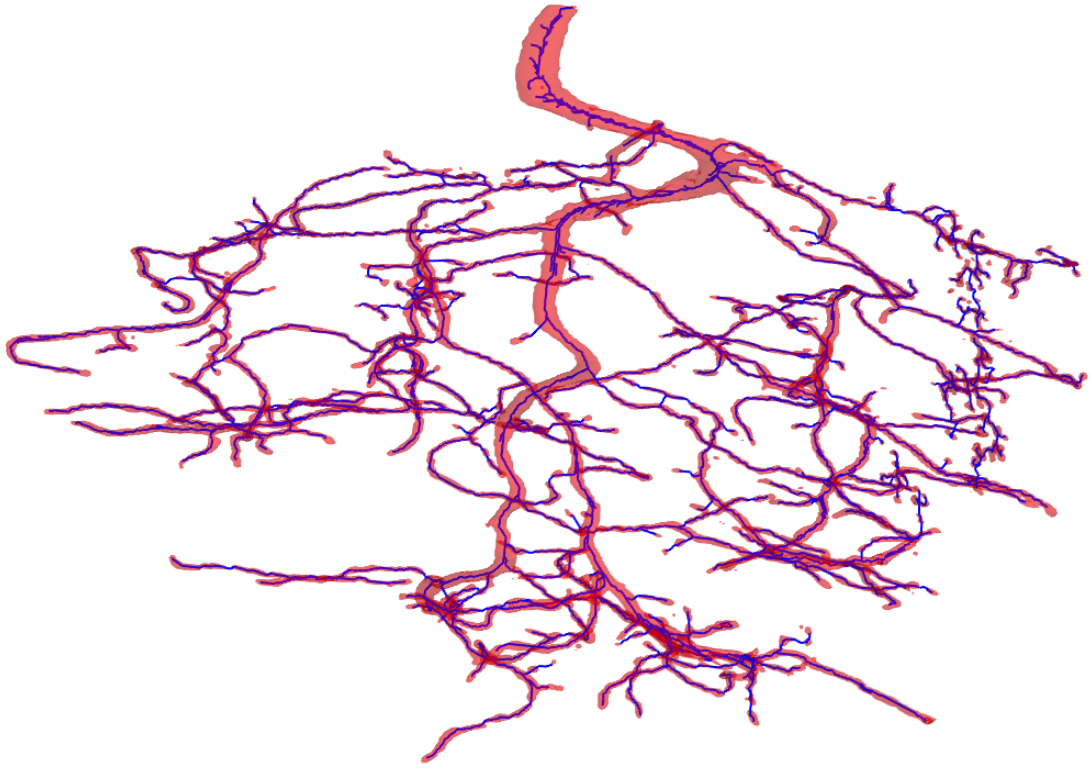


Figure 7.10: Extracted root model of Lupine 22 August. Majority of false positives are eliminated. Only small fraction of the MRI artifact plane remains as false positives.The algorithm assumes that the voxels are cubic. This causes the root structure to appear squeezed.

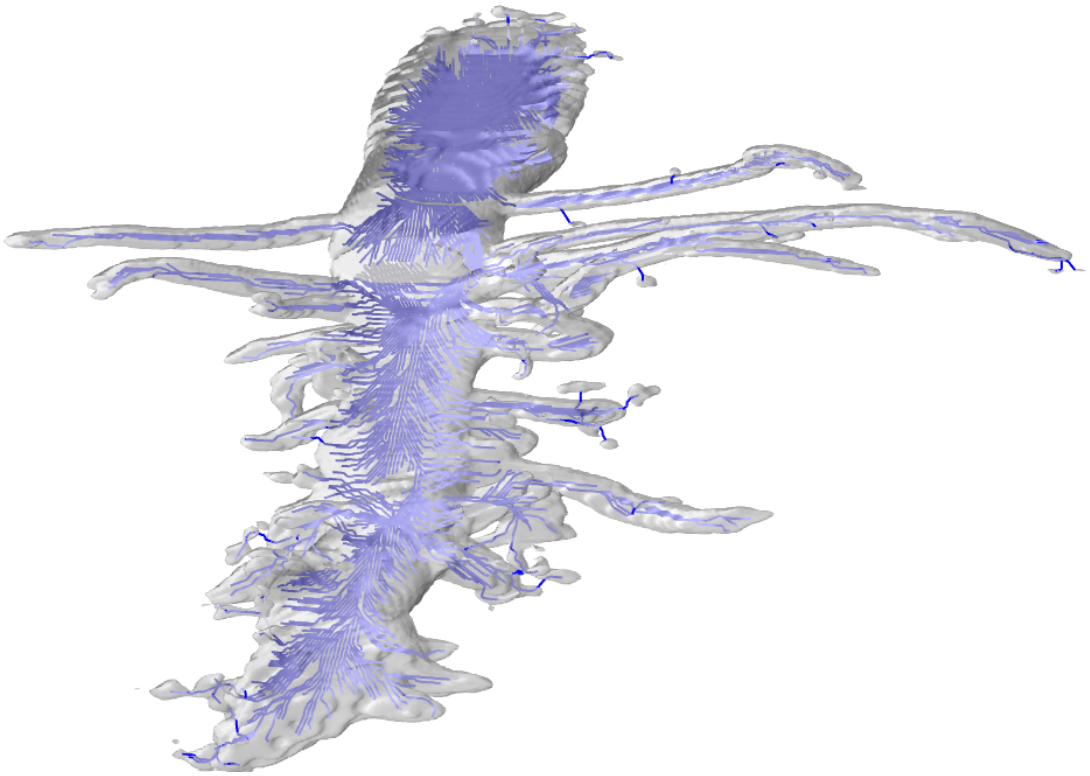


Figure 7.11: Extracted root model of Lupine Small. No false positives remain. An issue is duplicate detection of thick roots. This is due to the assumption of NMRooting that the root thickness is at most 3 voxel-wide.

8 Conclusion

This thesis investigated the automated segmentation of plant root MRI images as root vs non-root. The main contributions have been the data augmentation processes, mapping of 3D information to 2D RGB images and developing CNNs for super-resolution segmentations of the plant root MRI images. The validity of our approach has been investigated both qualitatively and quantitatively using synthetic and real data.

The random synthetic data has been continuously, iteratively improved as it has been seen that the accuracy of segmentation increases as the training samples are closer to reality. During the qualitative analysis of the segmentations, we have seen that most of the errors stem from the MRI artifacts rather than the noise found inside the soil. For future work, these MRI artifacts should be handled and incorporated into the data augmentation processes. Additionally, the extremely small loss values indicate that more variety in the root and soil noise structures are necessary.

We have seen that super-resolution is especially useful for MRI images with extremely thin roots as often the thickness of these roots are sub-voxel size in original resolution.

When compared with the manual reconstruction of the plant root MRI images, we see that our estimated segmentations are more precise in terms of root position and thickness. Some parts of the roots were missing in their annotations due to their hard to detect nature. Our method successfully detects these roots. Previously, root model extraction software has been unable to detect the root structures from the raw real data. While it still has its own problems, we have shown that upon segmentation of these MRI images, the algorithms can successfully detect and even complement the missing parts of the segmentation results of our method.

As an extension point, the future work can include the segmentation of data from other domains which make use of similar volume data. Examples may include medical applications such as brain MRI image segmentation for tumor detection.

Bibliography

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. (2016). "Tensorflow: a system for large-scale machine learning." In: *OSDI*. Vol. 16, pp. 265–283 (cit. on p. 11).

Ahrens, J., B. Geveci, and C. Law (2005). "Paraview: An end-user tool for large data visualization". In: *The visualization handbook* 717 (cit. on p. 20).

Behnke, S. (2001). "Learning iterative image reconstruction in the neural abstraction pyramid". In: *International Journal of Computational Intelligence and Applications* 1.04, pp. 427–438 (cit. on p. 16).

— (2018). "Private communications with Sven Behnke". In: (cit. on p. 7).

Carreira, J. and A. Zisserman (2017). "Quo vadis, action recognition? a new model and the kinetics dataset". In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, pp. 4724–4733 (cit. on p. 1).

Chawla, N. V., K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer (2002). "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16, pp. 321–357 (cit. on p. 17).

Chen, L.-C., G. Papandreou, F. Schroff, and H. Adam (2017). "Rethinking atrous convolution for semantic image segmentation". In: *arXiv preprint arXiv:1706.05587* (cit. on p. 15).

Dong, C., C. C. Loy, K. He, and X. Tang (2016). "Image super-resolution using deep convolutional networks". In: *IEEE transactions on pattern analysis and machine intelligence* 38.2, pp. 295–307 (cit. on p. 16).

Dusschoten, D. van, 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". In: *Plant physiology*, pp–01388 (cit. on pp. 1, 18, 51).

Fan, T. (2018). *Pytorch Implementation of Refinenet*. `https://github.com/thomasjpfan/pytorch_refinenet/` (cit. on p. 39).

Glorot, X. and Y. Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256 (cit. on pp. 11, 12).

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press (cit. on p. 11).

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680 (cit. on p. 16).

Hayat, K. (2017). "Super-resolution via deep learning". In: *arXiv preprint arXiv:1706.09077* (cit. on p. 5).

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (cit. on pp. 15, 32).

Higgins, D. M. (2003-2018). *How can we measure the signal-to-noise ratio (SNR)?* URL: `http://www.revisemri.com/questions/equip_qa/measuring_snr` (cit. on p. 25).

Horn, J. (2018). "Superresolution 3D Image Segmentation for Plant Root MRI". In: (cit. on pp. 9, 11, 18, 49).

Kingma, D. P. and J. Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (cit. on pp. 12, 13, 40).

Kleesiek, J., G. Urban, A. Hubert, D. Schwarz, K. Maier-Hein, M. Bendszus, and A. Biller (2016). "Deep MRI brain extraction: a 3D convolutional neural network for skull stripping". In: *NeuroImage* 129, pp. 460–469 (cit. on p. 16).

Kolesnik, M. and A. Fexa (2004). "Segmentation of wounds in the combined color-texture feature space". In: *Medical imaging 2004: Image processing*. Vol. 5370. International Society for Optics and Photonics, pp. 549–557 (cit. on p. 15).

Lai, W.-S., J.-B. Huang, N. Ahuja, and M.-H. Yang (2017). "Deep laplacian pyramid networks for fast and accurate superresolution". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. 3, p. 5 (cit. on p. 16).

Ledig, C., L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. (2017). "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network." In: *CVPR*. Vol. 2. 3, p. 4 (cit. on p. 16).

Lin, G., A. Milan, C. Shen, and I. D. Reid (2017). "RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation." In: *Cvpr*. Vol. 1. 2, p. 5 (cit. on pp. 1, 15, 31–33).

Long, J., E. Shelhamer, and T. Darrell (2015). "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440 (cit. on p. 15).

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". In: (cit. on pp. 11, 38).

Pereira, S., A. Pinto, V. Alves, and C. A. Silva (2016). "Brain tumor segmentation using convolutional neural networks in MRI images". In: *IEEE transactions on medical imaging* 35.5, pp. 1240–1251 (cit. on p. 16).

Perlin, K. (1985). "An image synthesizer". In: *ACM Siggraph Computer Graphics* 19.3, pp. 287–296 (cit. on p. 23).

Pflugfelder, D., R. Metzner, D. Dusschoten, R. Reichel, S. Jahnke, and R. Koller (2017). "Non-invasive imaging of plant roots in different soils using magnetic resonance imaging (MRI)". In: *Plant methods* 13.1, p. 102 (cit. on p. 1).

Pham, C.-H., R. Fablet, and F. Rousseau (2017). "Multi-scale brain MRI super-resolution using deep 3D convolutional networks". In: (cit. on p. 16).

Rahman, M. A. and Y. Wang (2016). "Optimizing intersection-over-union in deep neural networks for image segmentation". In: *International Symposium on Visual Computing*. Springer, pp. 234–244 (cit. on pp. 17, 38).

Ren, S., K. He, R. Girshick, and J. Sun (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*, pp. 91–99 (cit. on p. 1).

Ronneberger, O., P. Fischer, and T. Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241 (cit. on p. 15).

Schaul, T., S. Zhang, and Y. LeCun (2013). "No more pesky learning rates". In: *International Conference on Machine Learning*, pp. 343–351 (cit. on p. 12).

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". In: (cit. on p. 1).

Schroeder, W. J., B. Lorensen, and K. Martin (2004). *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware (cit. on p. 20).

Schroff, F., A. Criminisi, and A. Zisserman (2008). "Object Class Segmentation using Random Forests." In: *BMVC*, pp. 1–10 (cit. on p. 15).

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." In: *VISAPP (2)*, pp. 24–33 (cit. on pp. 1, 17, 18, 43).

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". In: *International Workshop on Multiple Classifier Systems*. Springer, pp. 82–91 (cit. on p. 17).

Thai-Nghe, N., Z. Gantner, and L. Schmidt-Thieme (2010). "Cost-sensitive learning methods for imbalanced data". In: *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, pp. 1–8 (cit. on p. 17).

Xie, S., R. Girshick, P. Dollár, Z. Tu, and K. He (2017). "Aggregated residual transformations for deep neural networks". In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, pp. 5987–5995 (cit. on p. 1).

Yosinski, J., J. Clune, Y. Bengio, and H. Lipson (2014). "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*, pp. 3320–3328 (cit. on p. 31).

Zikic, D., Y. Ioannou, M. Brown, and A. Criminisi (2014). "Segmentation of brain tumor tissues with convolutional neural networks". In: *Proceedings MICCAI-BRATS*, pp. 36–39 (cit. on p. 16).