Machine Learning and Deeplearning for Dummies

Aryeh Weiss
Faculty of Engineering
Bar Ilan University

Anne Alerding
Department of Biology
Virginia Military Institute

Presented at:

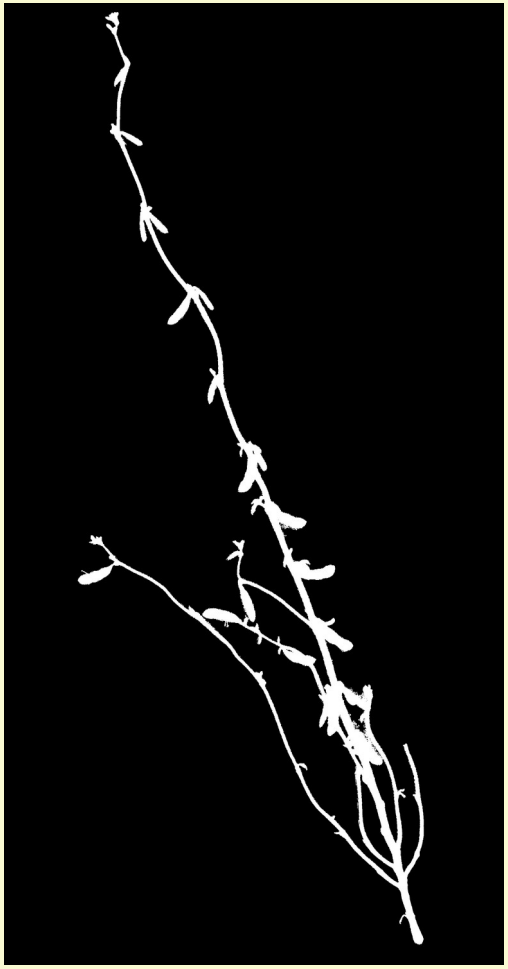
1. Classical Image Processing
2. “Shallow” machine learning
3. Deep learning

Classical Image Processing

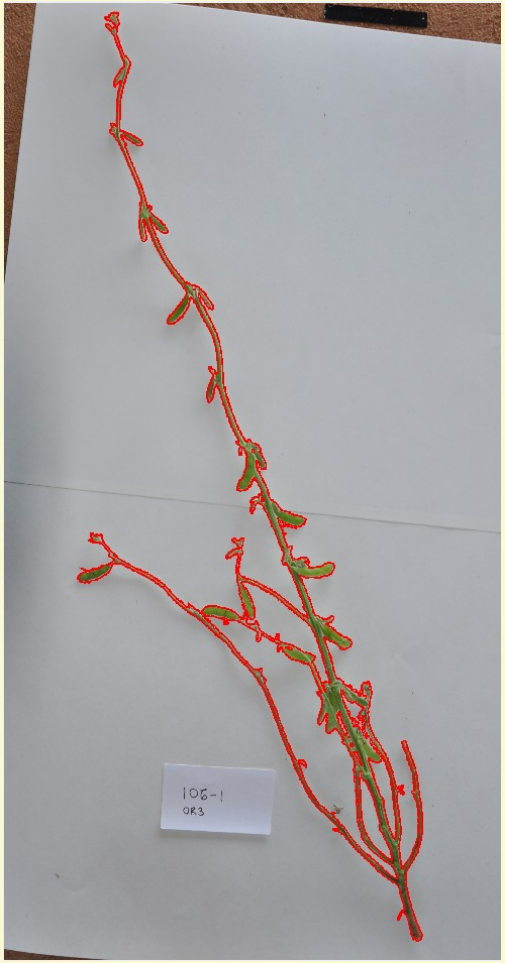
- Noise reduction and background leveling
- Filtering to enhance objects vs background
- Thresholding to identify objects
- Analysis based on measured properties

Example – segmentation of whole plant image

Original Image



Segmentation



Overlay of segmentation Edges

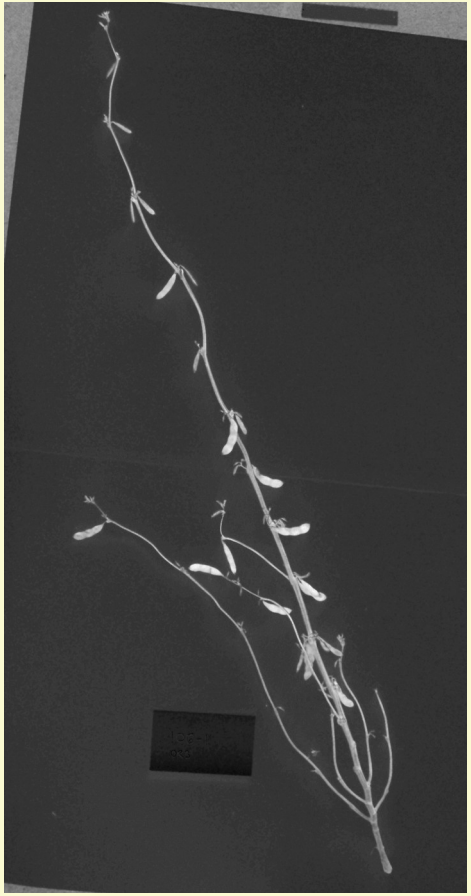
Elements of the workflow

- Conversion to LAB colorspace
- Thresholding on intensity and color dimensions
- Size criteria to eliminate small artifacts
- Morphological operations

Example – conversion to LAB colorspace

Luminence

a-channel



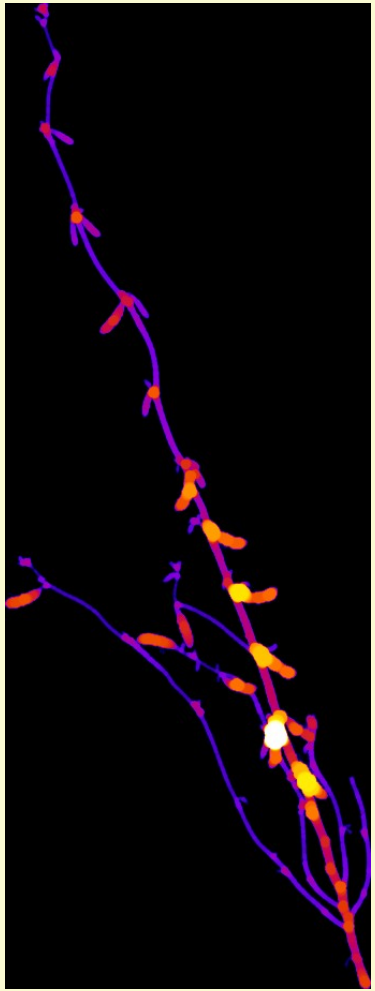
b-channel

Limitation – how to segment pods from stems

The plant is very different from the background
– color, brightness, distinctive shape (morphology)

Pods look very much like stems.
– Thickness?
  – Main stem is thick
  – Some pods are thin
– Color? varies from brown to green

Example – heat map of minimum thickness



Heat map of minimum thickness at each point of a whole plant image.

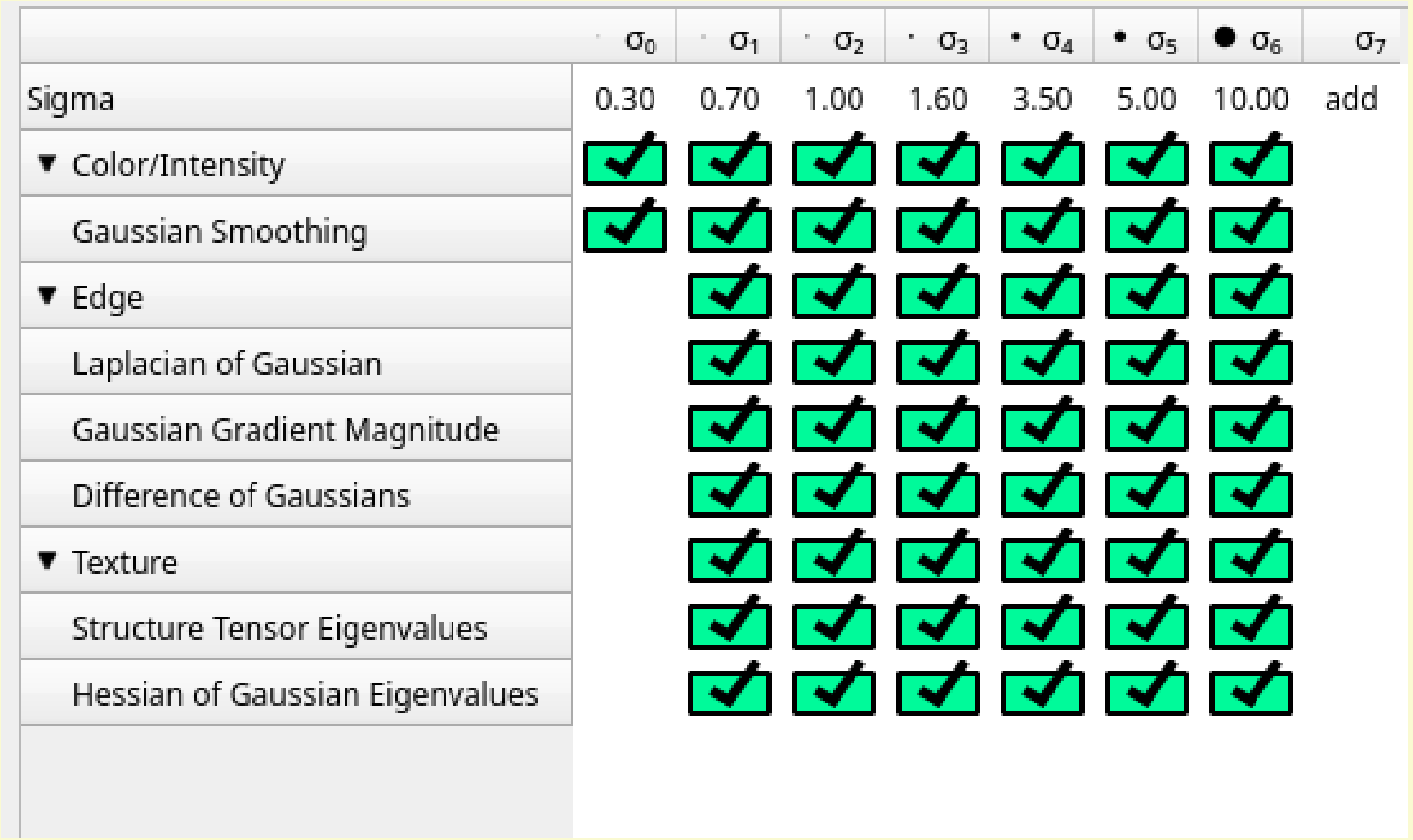
Pods are generally thick than stems, but the main stem is thicker that some pods.

Machine Learning

- Measure N features of each pixel.
- Create a N dimensional feature space.
- Cluster the points to minimize some loss function.

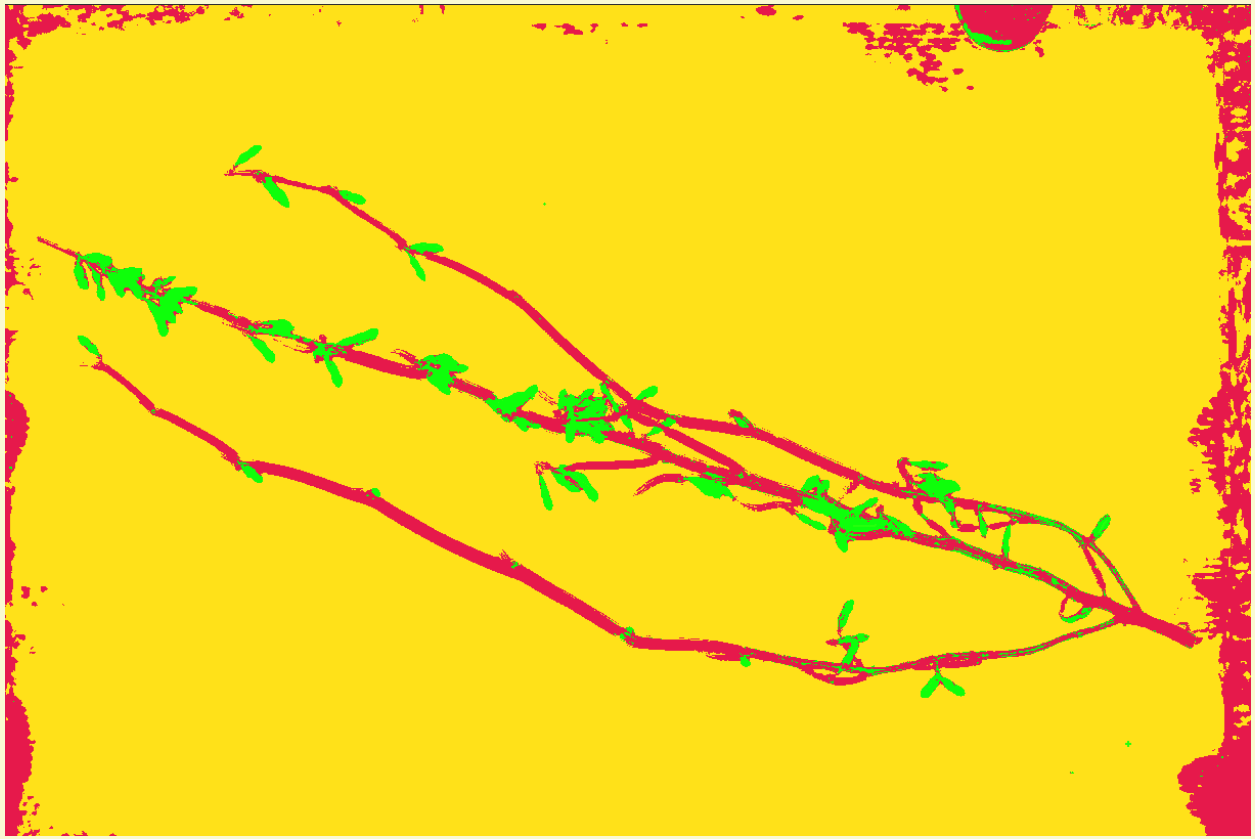
Examples:
– Ilastik
– Trainable Weka Segmentation (ImageJ/Fiji)

Ilastik – feature selection



Ilastik – sparse labeling

Ilastik - prediction



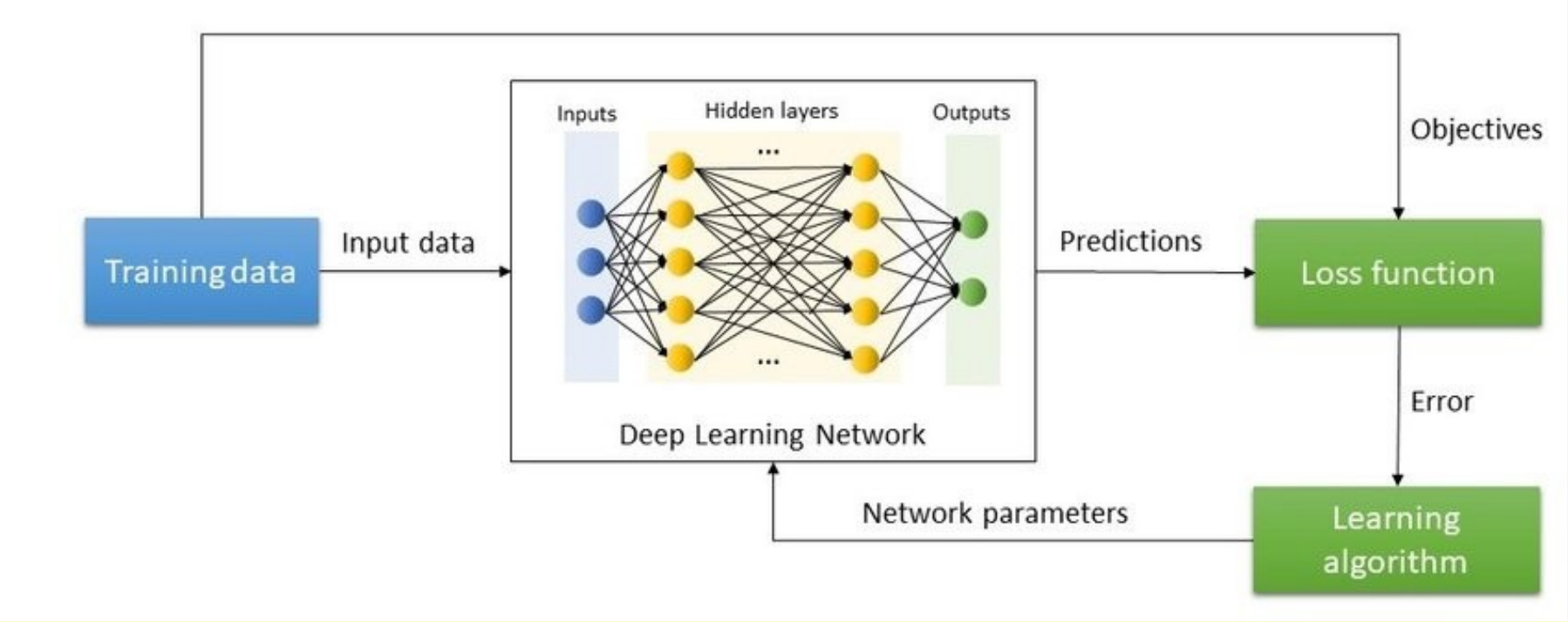
Machine Learning

- Provides convenient tools for high dimensional feature spaces.
- Provides decent results with relatively little annotation.
- Requires prior selection of feature space.
- When it works, it is reasonably simple to know what it did.

Deep Learning – based on neural networks

- Finds the relevant features as part of the training.
- Computationally intensive (especially for images)
- Python libraries highly developed
  – Tensorflow, Pytorch
- Prepared programs – eg ZeroCOSTDL4Mic
- GUI tools – eg mianalyzer, MATLAB Deeplearning toolbox
- Can require a large training set.

Neural networks



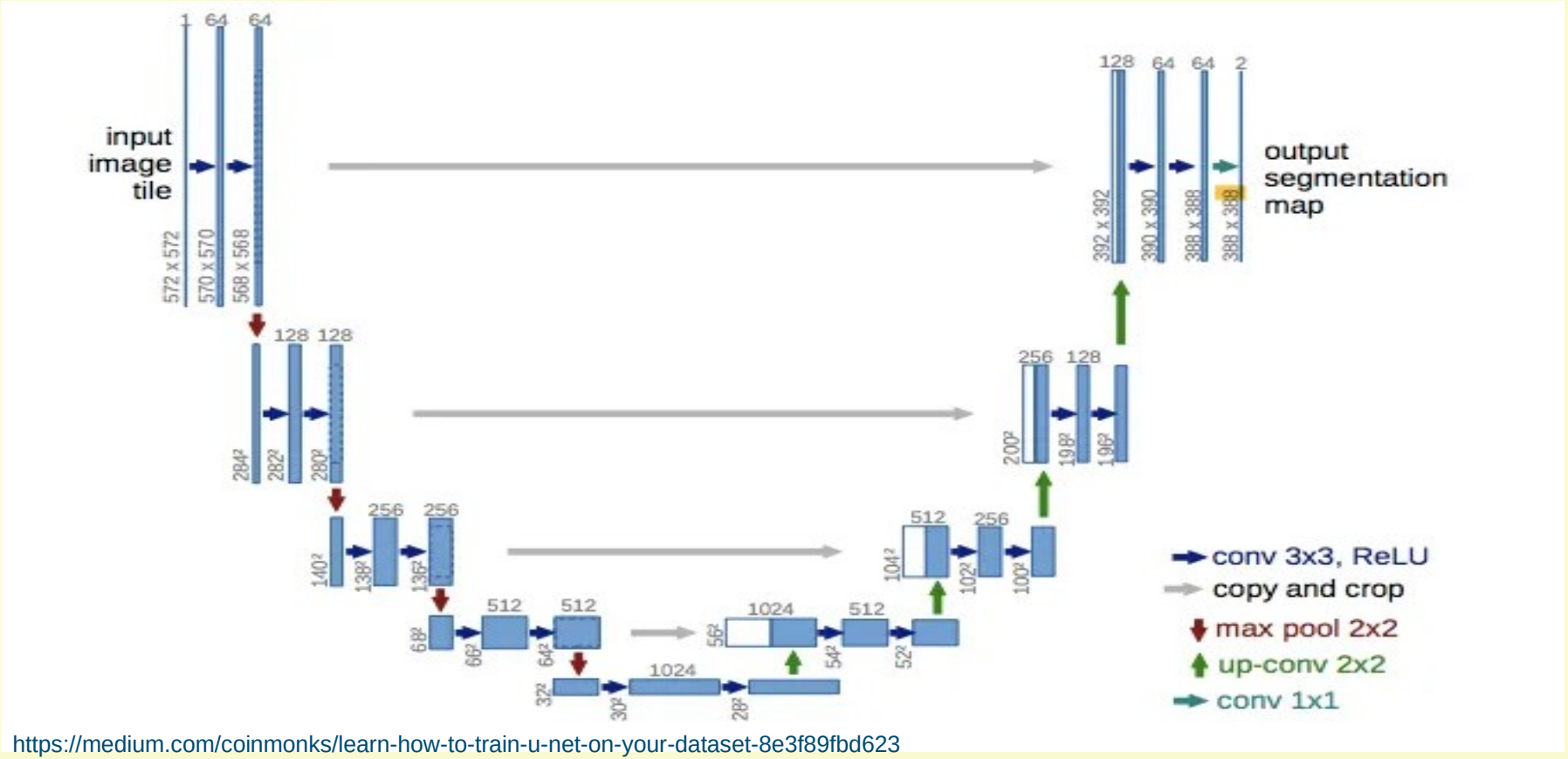
- Many trainable parameters (can easily be millions)
- Training requires serious computing power (GPU)

https://www.researchgate.net/figure/Training-process-in-Deep-Learning-algorithms_fig1_350361033

Convolutional NN (CNN) and Unet

- Images often contain millions of pixels. A fully connected NN to process them by “brute force” would be huge, even for current computer systems.
- The Unet architecture + CNNs allow semantic segmentation with a reasonably sized network.

Unet Architecture

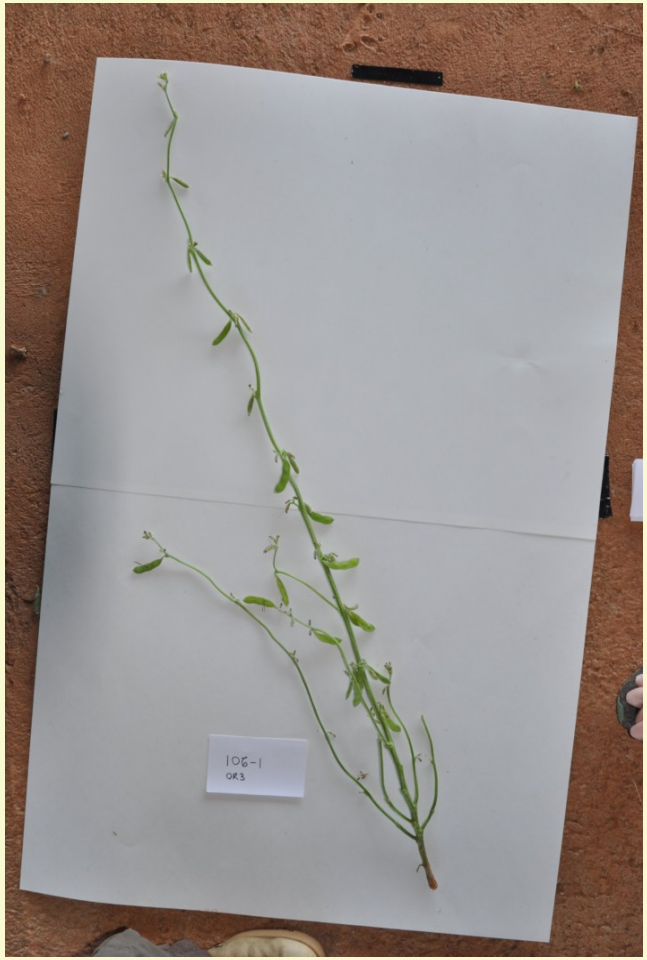


https://medium.com/coinmonks/learn-how-to-train-u-net-on-your-dataset-8e3f89fbd623

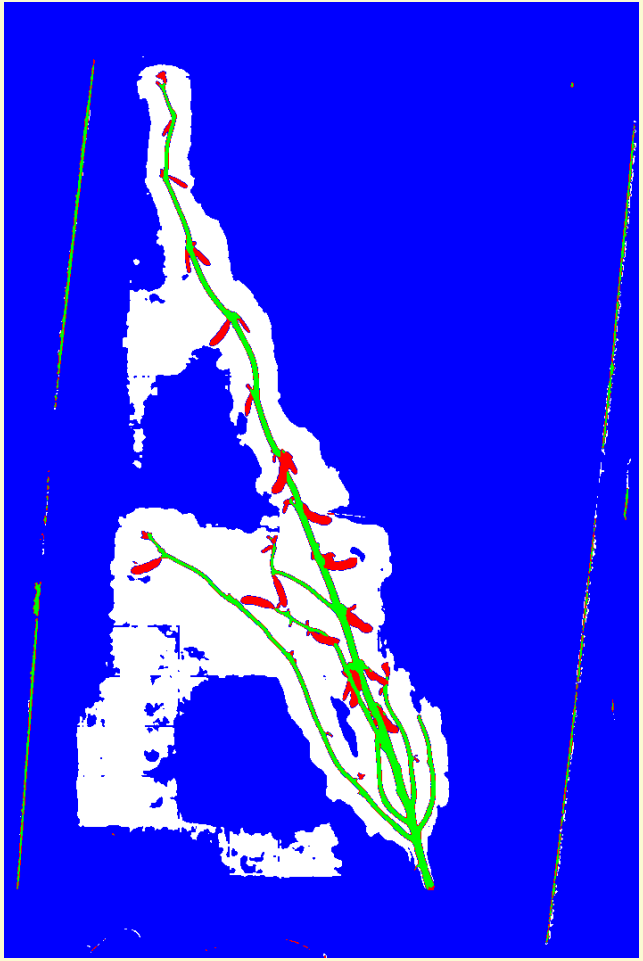
Why can dummies do it?

- There are many free software tools (libraries, applications)
  Tensorflow, Pytorch, Mianalyzer (GUI), DeepImageJ.
  Microscopy Image Browser (MIB)
- Serious computing power is readily available (GPU clusters, cloud computing). My laptop has an Nvidia Ti3080 with 16GB DDR6 RAM. (Gets pretty hot!)
- On-line tutorials and free courses are available.
- **Most important – an incredible community of developers and users who provide help in close to real time. (image.sc and more)**

Semantic segmentation of pods and stems



Original image



LUT: white (0): unlabeled; blue (1): background; green (2): stem; red (3): pod

NN architecture: UNET
Backbone: efficientnet b1

Training set: 30 annotated images.

4 classes:
- pod
- stem
- background
- unlabeled

Trained for 100 epochs

Used mianalyzer (headless)

Deep learning workflow

- Prepare a training set of representative images.
- Annotate the images (this is the bottleneck!)
- Convert the annotated images to the correct format.
- train (training set, validation set)
- test (test set – not to be used during training)
- prediction (naive images)

Annotation

- Many tools for labeling (Labelme, ImageJ/Fiji, QuPath)
- Manual labeling , Semi-automatic labeling, “Magic wand”
- Semantic segmentation (pixel classification).
  Requires fewer images, but annotation is harder.
- Instance segmentation (classify objects or images)
  Requires many images. (Example – MNIST digits)

Annotation with QuPath

Example of an annotated image.

Why QuPath?

1. Because it has a convenient “magic wand” annotation mode that works nicely, and also a nice mode for erasing over filling.

2. Because it has a scripting capability for converting annotated images into labeled image files.

Annotated Images

- Label images: set pixel values according to class

QuPath Groovy script

```
def imageData = getCurrentImageData()

// Define output path (relative to project)
def outputDir = buildFilePath(PROJECT_BASE_DIR, 'export')
mkdirs(outputDir)
def name = GeneralTools.getNameWithoutExtension(imageData.getServer().getMetadata().getName())
def path = buildFilePath(outputDir, name + "-labels.tif")

// Define how much to downsample during export (may be required for large images)
double downsample = 1

// Create an ImageServer where the pixels are derived from annotations
def labelServer = new LabeledImageServer.Builder(imageData)
  .backgroundLabel(0, ColorTools.WHITE) // Specify background label (usually 0 or 255)
  .downsample(downsample)    // Choose server resolution; this should match the resolution at which tiles are exported
  .addLabel('bg', 1)      // Choose output labels (the order matters!)
  .addLabel('pod', 2)
  .addLabel('stem', 3)
  .multichannelOutput(false) // If true, each label refers to the channel of a multichannel binary image (required for multiclass probability)
  .build()

// Write the image
writeImage(labelServer, path)
```

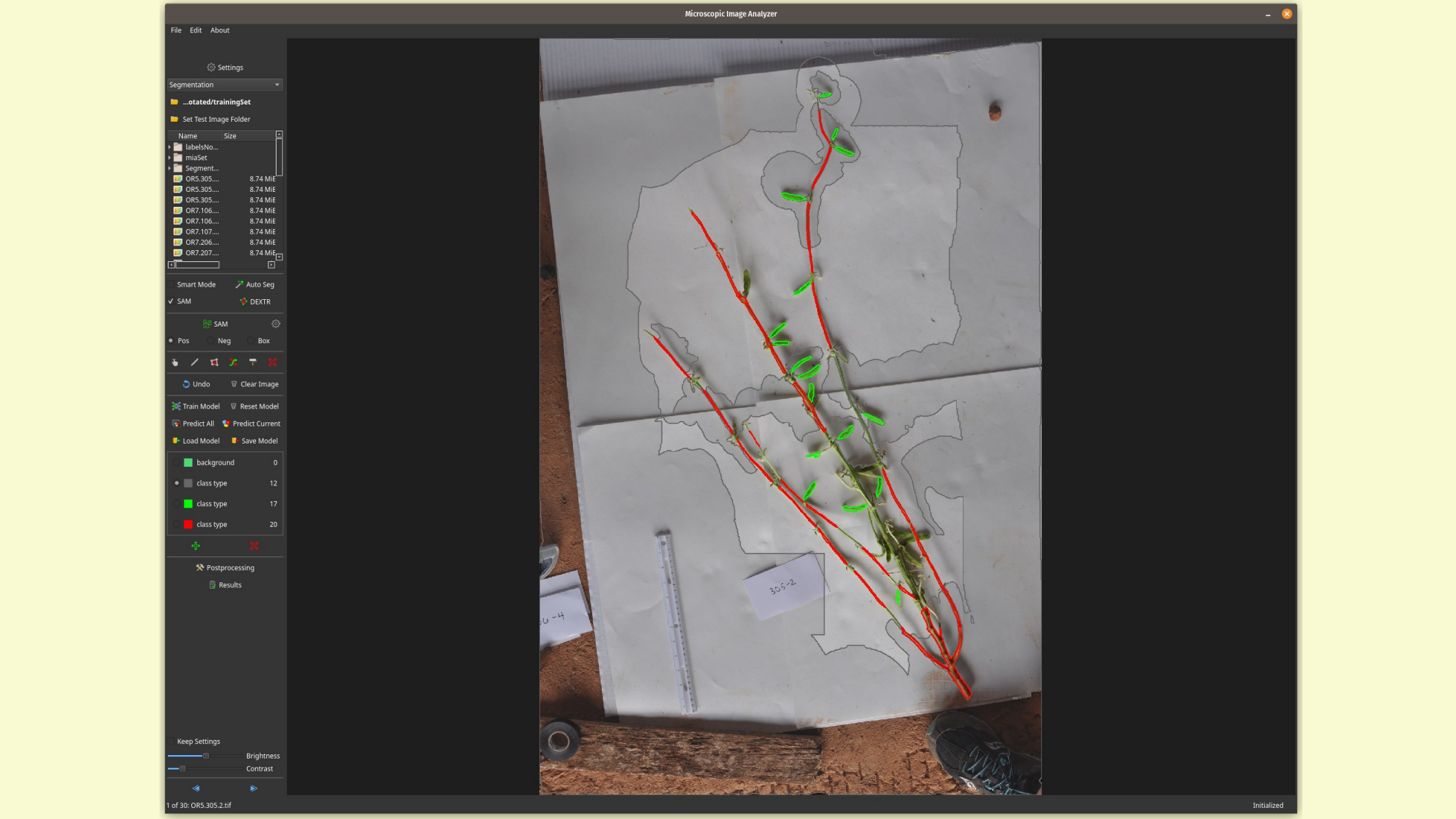
mianalyzer for semantic segmentation

mianalyzer is a GUI application that allows the user to choose a NN architecture and backbone, set training parameters, train and predict.

Written by Nils Koerber (who is very helpful)
https://www.biorxiv.org/content/10.1101/2022.01.14.476308v1

Help is available on image.sc , tag mia

We will describe an mianalyzer based workflow.


Microscopic Image Analyzer
File
Edit
About
Settings
Segmentation
...otated/trainingSet
Set Test Image Folder
Name
Size
labelsNo...
miaSet
Segment...
OR5.305....
OR5.305....
OR5.305....
OR7.106....
OR7.106....
OR7.107....
OR7.206....
OR7.207....
8.74 MiE
Smart Mode
Auto Seg
SAM
DEXTR
SAM
Pos
Neg
Box
Undo
Clear Image
Train Model
Reset Model
Predict All
Predict Current
Load Model
Save Model
background
0
class type
12
class type
17
class type
20
Postprocessing
Results
Keep Settings
Brightness
Contrast
1 of 30: OR5.305.2.tif
Initialized

Labeled images and NPZ files

- mianalyzer requires NPZ files that encode contours.
- Nils Koerber wrote a script that converts label-images to the NPZ files that mianalyzer requires for training.
- The result is a set of NPZ files that provide the ground truth.
  These will reside in a subdirectory called Segmentation_labels.

A multiclass conversion script to convert label images to NPZ images can be found here:
https://github.com/aryehw/miaHeadless/tree/master/mia/multiclass_conversion

The labeling workflow:
– Annotate images in QuPath
– Save annotated images as pixel-labeled images
– Convert labeled images to NPZ

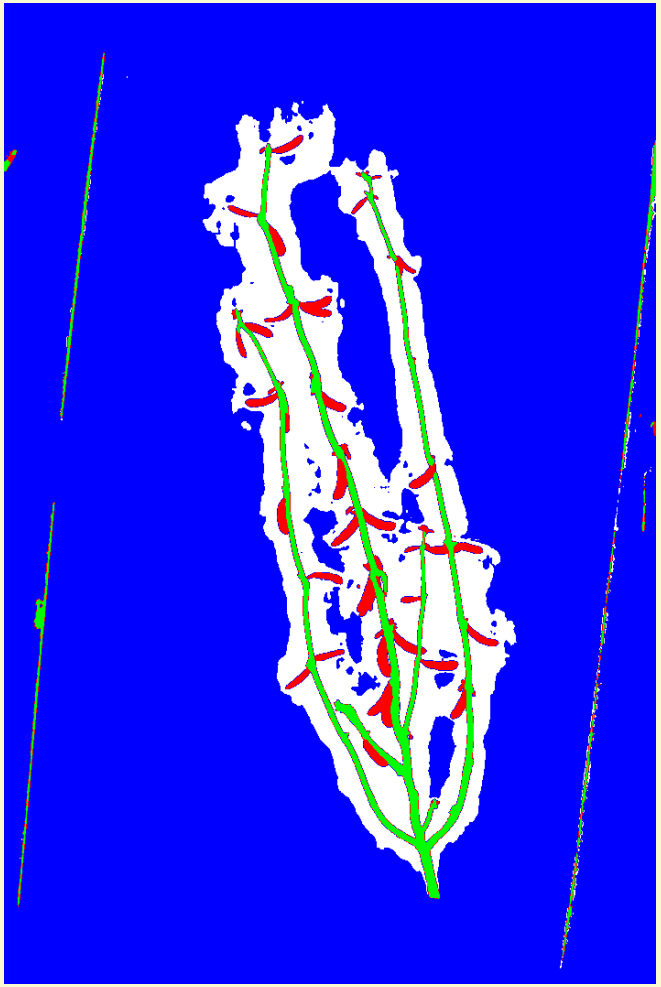
Training and Prediction

- Select the training directory.
- Select NN architecture, backbone, epochs, learning rate, etc.
- Train
- Predict
- Save predictions and model.

Remote operation

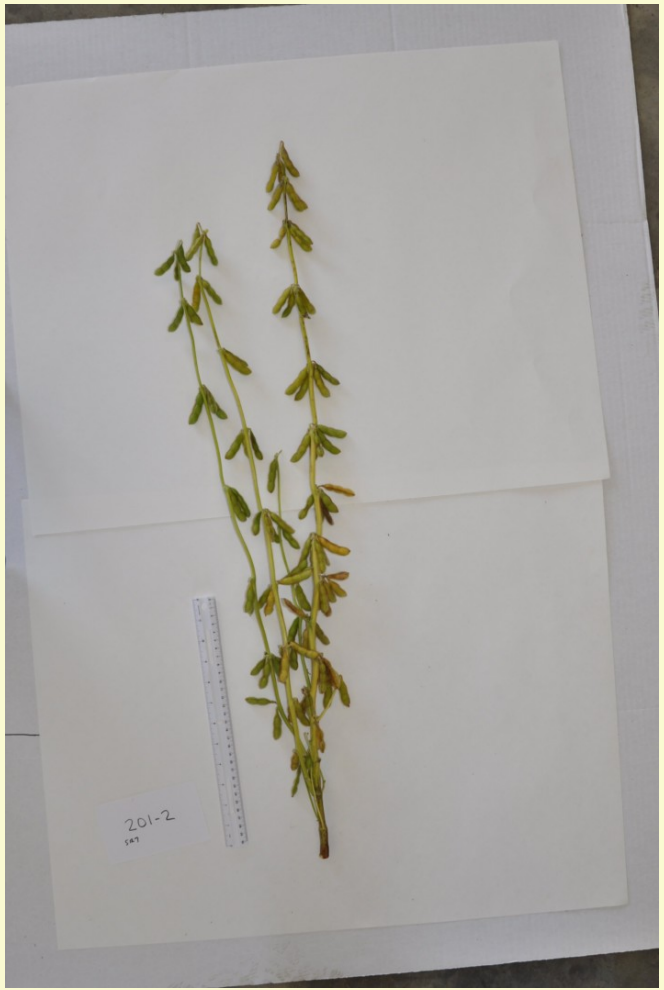
- NN training is very compute intensive. It requires a GPU with at least 12GB of RAM (for my problem)
- mianalyzer is a GUI application, which uses the X-windows system. This is too slow to run over a remote network (screen update is too slow).
  However, mianalyzer was not meant to run headless.
- Nils Koerber (author of mianalyzer) provided a basic script that allows headless operation of the training phase.
- We adapted this script to carry out training and prediction from a Python script.
  This script is available on a Github site:
  https://github.com/aryehw/miaHeadless/tree/master/mia
- We can also run the training on a Lenovo P1 Gen 5 Extreme laptop computer with a Nvidia Ti3080 GPU with 16 GB DDR6 RAM. A cooling stand is recommended for long training runs.

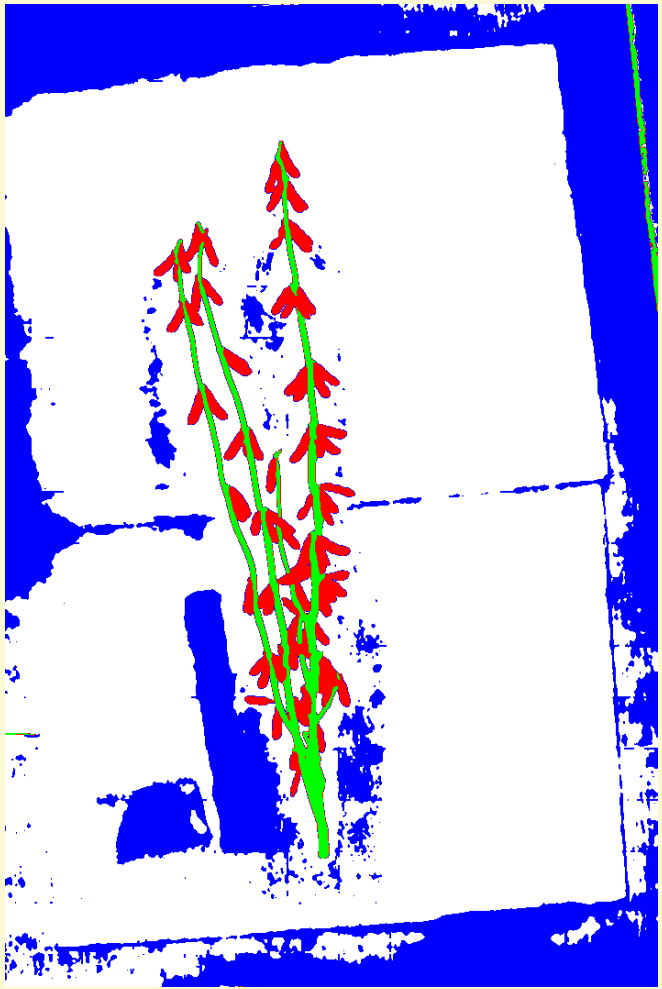
Results – Unet with efficientnetb1



Harvest 1 OR5 9 Aug 17

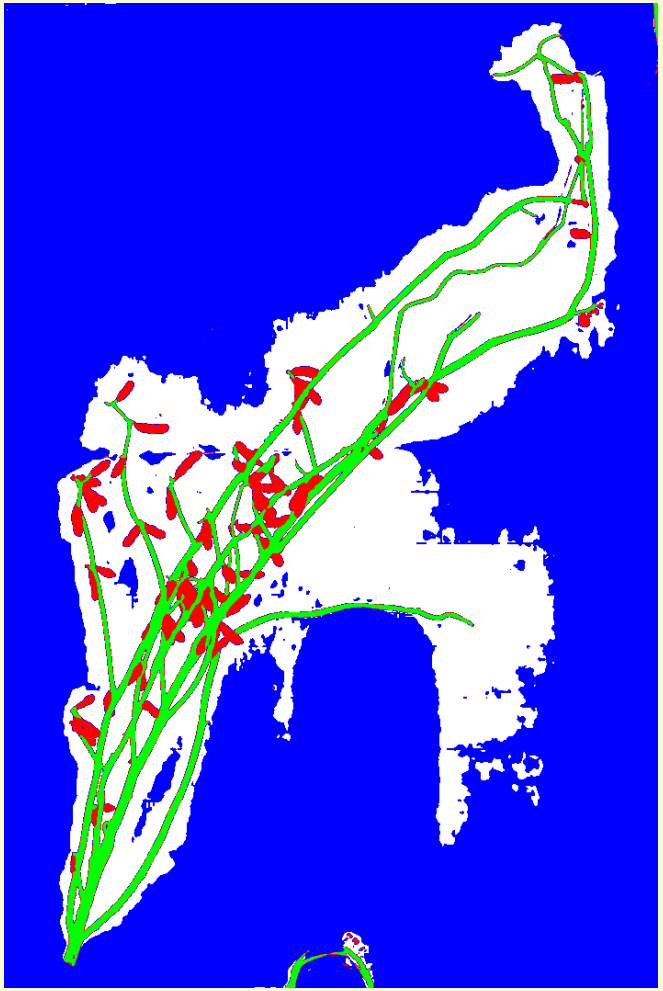
Results – Unet with efficientnetb1





Harvest 4 SR7 6 Oct 17

Results – Unet with efficientnetb1

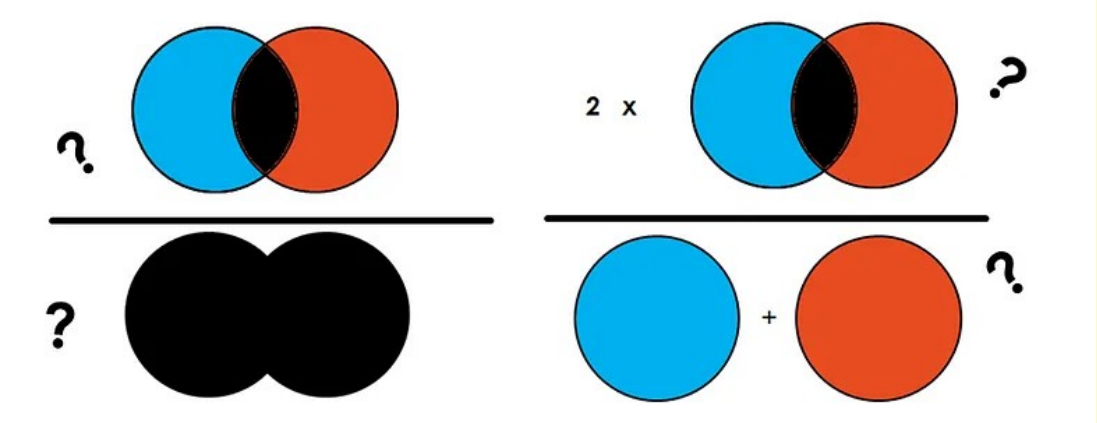


Harvest 8 OR7 5,7 Oct 17

Evaluation of Results

Intersection over Union (IoU)

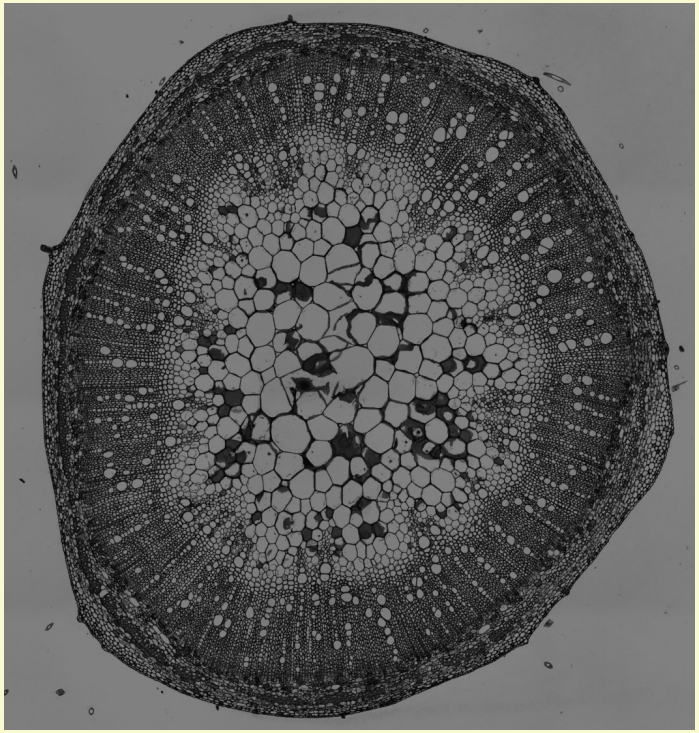
Dice coefficient



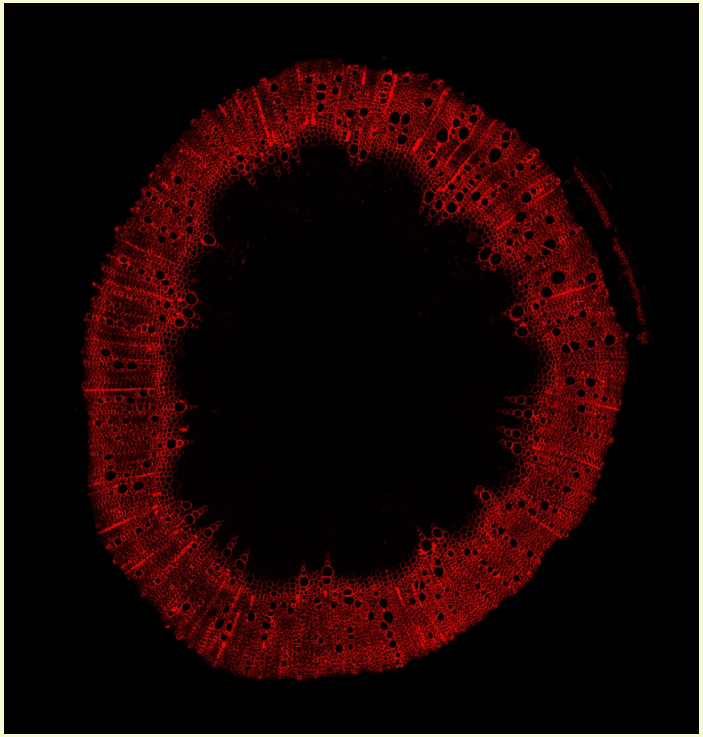
- Both of these metrics range from 0 (completely wrong) to 1 (completely correct)
- An annotated test set is required to calculate the metrics.
- There are other metrics (precision, accuracy, etc). These are probably better for classification problems.

https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

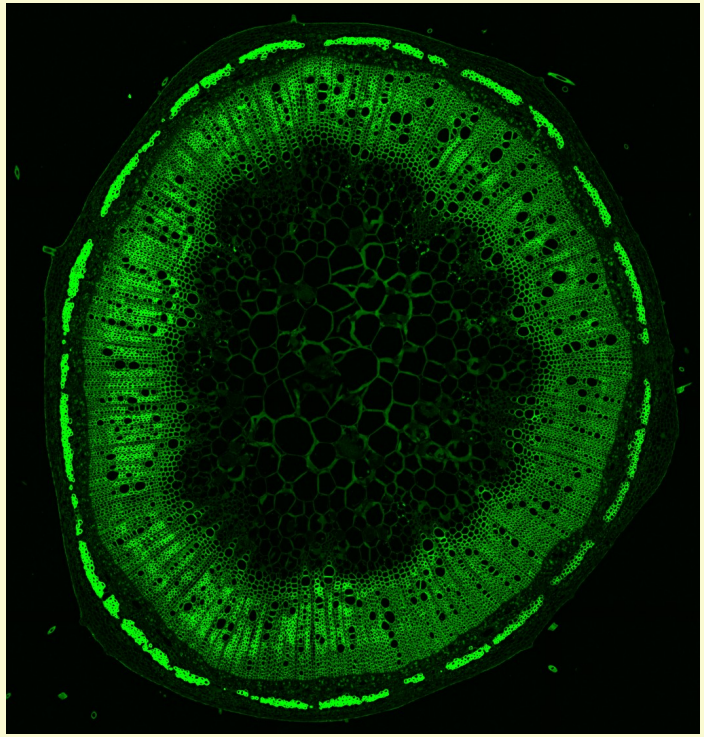
More segmentation problems



Transmitted light

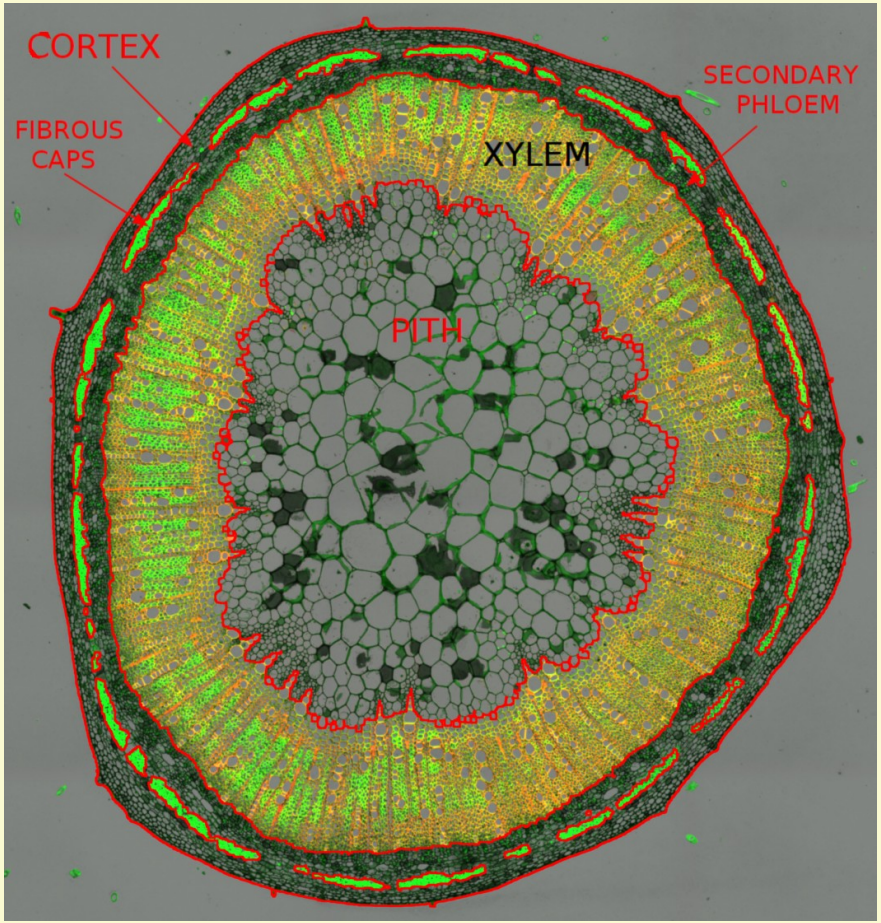


Xylem
(safranin)



Cellulose
(fast green)

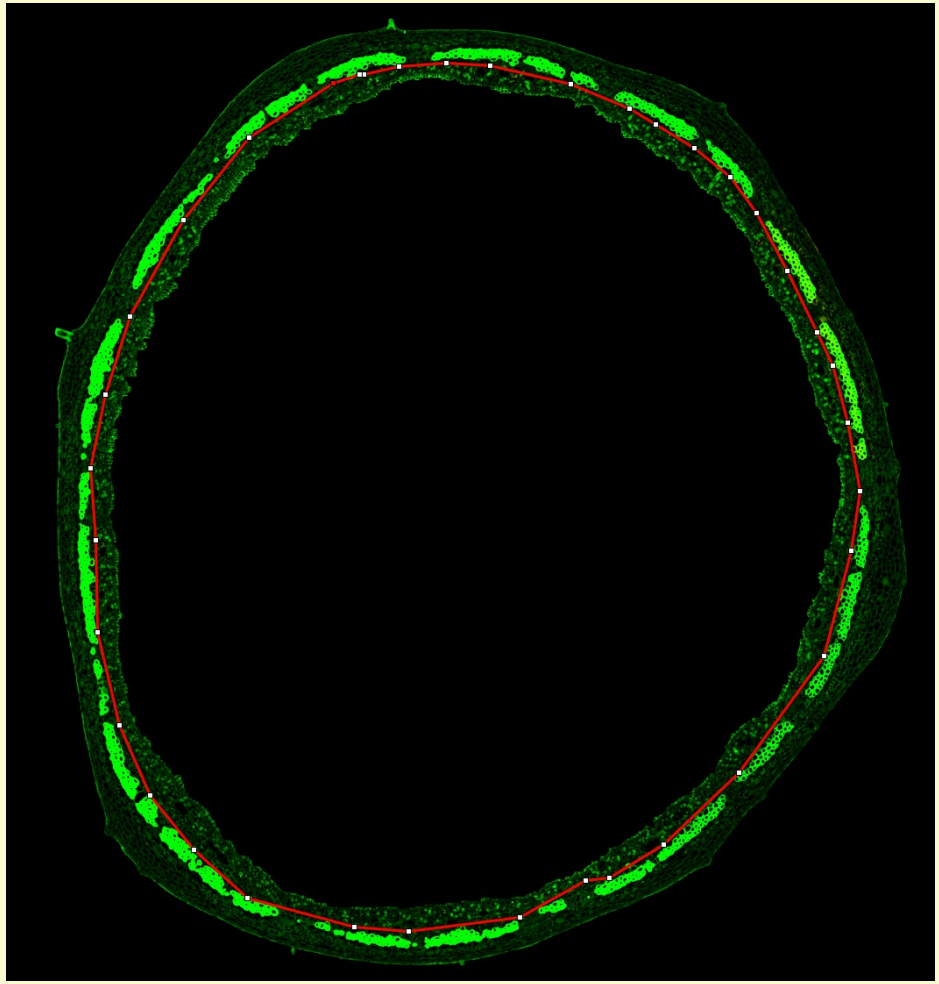
Segmentation of stem section (classical)



Segmented stem section.
Pith, Xylem, Phloem caps.

Problem: segment secondary phloem

Problem – isolation of secondary phloem

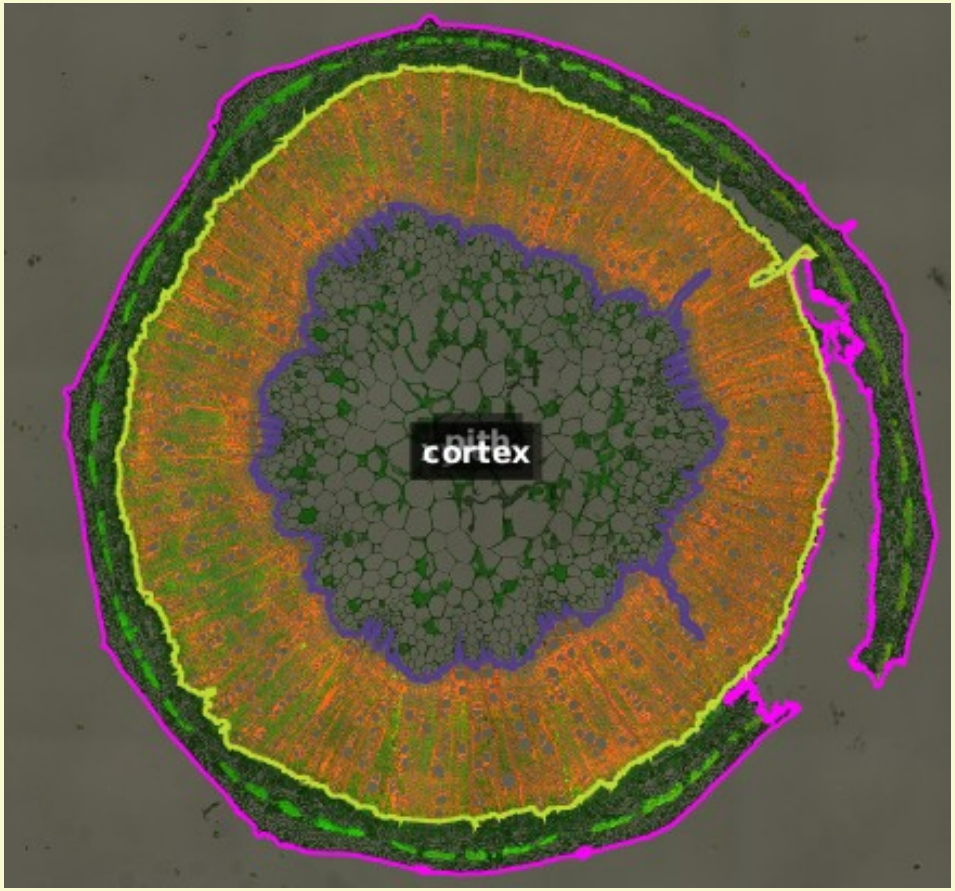


We want to detect the gelatenous fibers in the secondary phloem (the area inside of the red boundary)

This is a candidate for deep learning. We need a set of annotated images.

Most of the tissues can be automatically annotated, but the secondary phloem must be manually labeled.

Problem – damaged sections

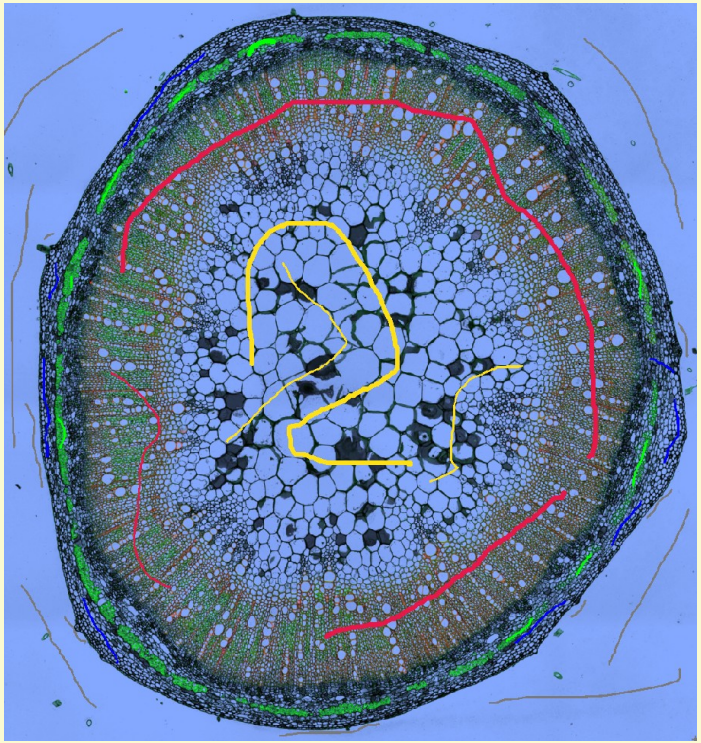


The classical image processing approach is not robust against damage sich as breaks in the cortex.

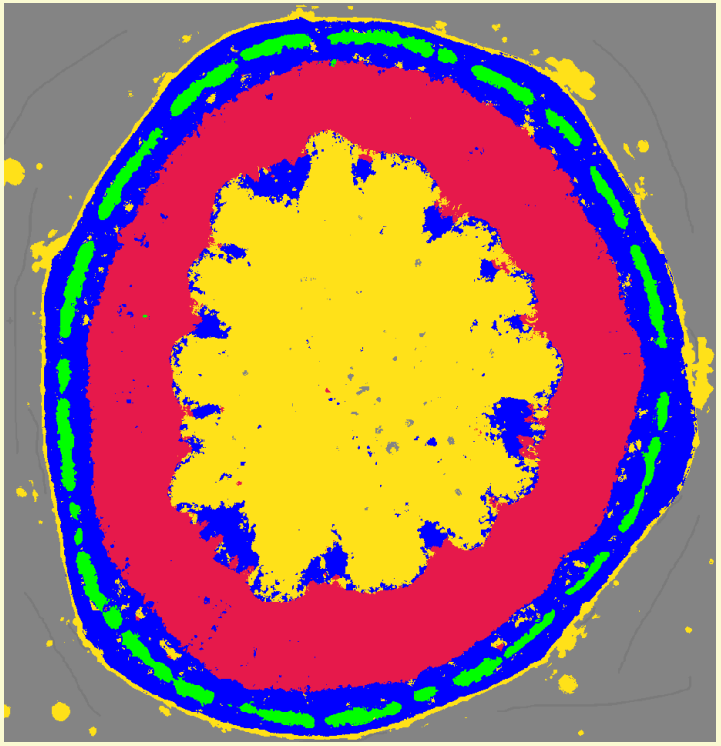
Fibrous phloem was not segmented.

Machine learning could probably recognize the fibrous caps, even if trained on undamaged sections.

Segmentation with Ilasktik (“shallow” ML)



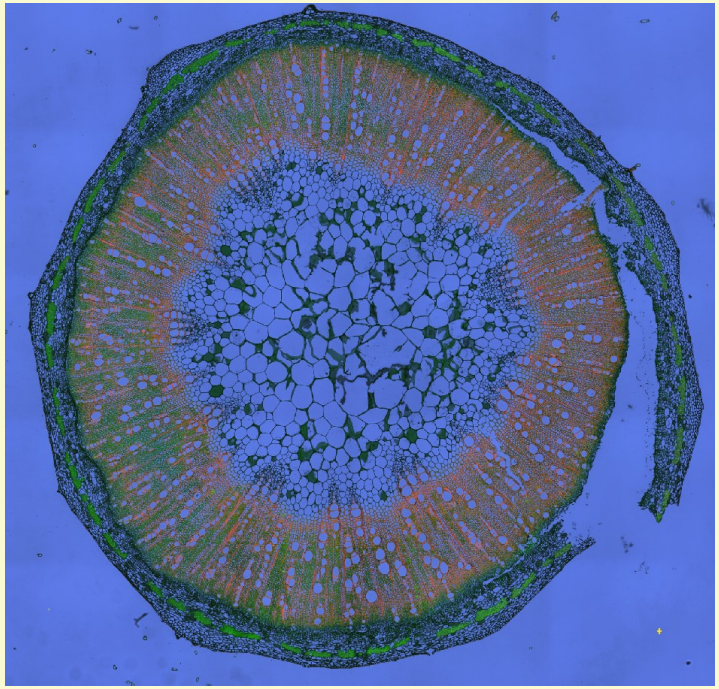
Labeled image



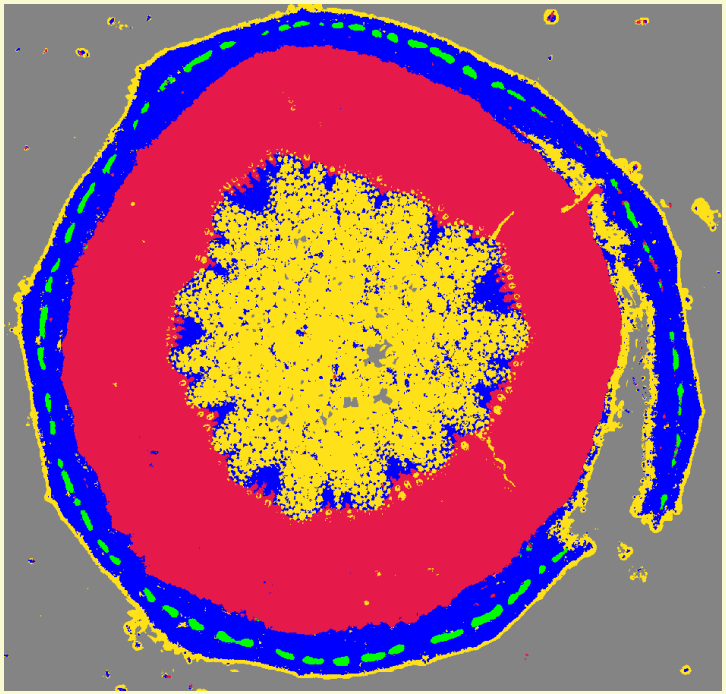
predictions

yellow:	pith
red:	xylem
blue:	cortex+phloem
green:	fibrous phloem
gray:	background

Ilasktik on unlabeled image



Naive image

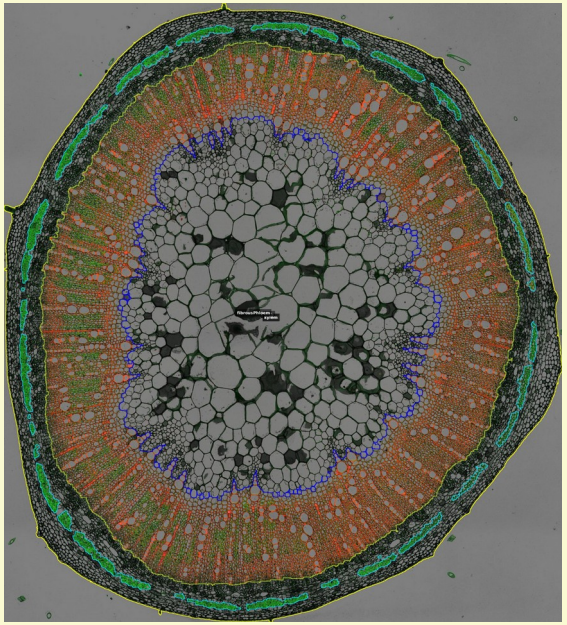


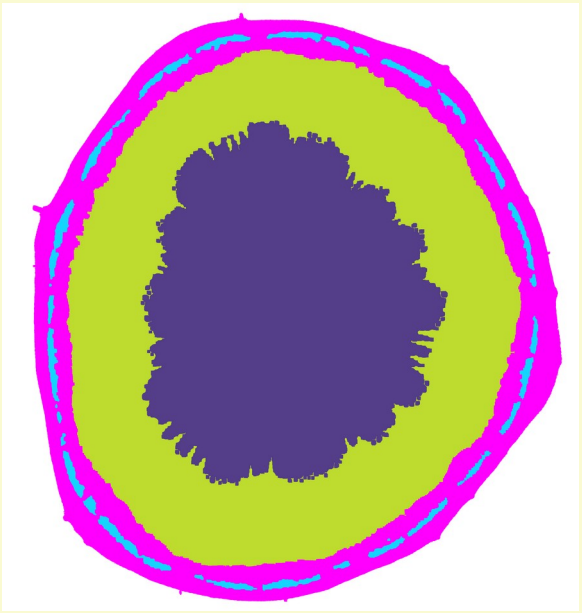
predictions

yellow: pith
red: xylem
blue: cortex+phloem
green: fibrous phloem
gray: background

Based on 4 sparsely labeled images. Not bad, especially for the fibrous phloem. However, it cannot distinguish primary xylem from cortex, because spatial location is not part of the feature space. Post processing could resolve the false assignments.

Using QuPath to fully annotate for DL





Segmented image with ROIs

label-image

Workflow:

1. Segment xylem, pith, cortext and fibrous phloem (where possible) with ImageJ/Fiji python script.
2. Import ImageJ ROIset into QuPath – regions are then annotated.
3. Add annotations for objects that the Fiji script does not segment.
4. Use QuPath groovy script to export label images

Summary (1)

- Classical image processing can segment some plant tissues very well.
- Machine learning allows one to use a large feature space to segment difficult cases.
- Deeplearning can be used for even more difficult cases where the feature space is not obvious.
- Classical image processing can clean up the output of the semantic segmentation from ML and DL.

Summary (2)

A rich set of tools enable ML and DL to be done by dummies.

- ML: Trainable Weka segmentation and Ilastik (and more)
- DL: Tensorflow, pytorch, mianalyzer, DeepImageJ, Deep Learning Toolbox (MATLAB), Microscopy Image Browser (and more).
- I have focused on ImageJ/Fiji, Ilastik, QuPath, and Mianalyzer as tools for image processing, shallow ML, annotation, and DL.

There are many other tools available – these are the tools that this dummy found easiest to use.

Addendum

Since preparing the above slides, I learned of the Microscopy Image Browser (MIB). This is a rich processing environment that includes DL tools.

- Runs in MATLAB or standalone.
- Includes preprocessing, training, and prediction tools. Requires no programming.
- Website: http://mib.helsinki.fi/
- Author: Ilya Belevich. Provides amazing help, especially via image.sc forum (tag is mib)