ADS-AI

DataLab Tasks

This page describes the DataLab tasks, deliverables, and their connection to the ILOs. By completing these tasks, you will produce a set of deliverables that align with the ILOs. In the following table, you will find a summary of all the DataLab tasks.

Task	ILO	Description	Timeline
1 Image Annotation and Review	ILO 6.4A, ILO 8.5 prerequisite	Annotate an image to create a labeled dataset for training & peer review	W1 - W3
2 Petri Dish Detection and Extraction	ILO 6.4B	Detect and extract the Petri dish region from images	W1 onwards
3 Plant Instance Segmentation	ILO 6.4C	Segment plant instances	W1 onwards
4 Dataset Preparation for Model Training	NA	Prepare images and labels for model training	W2 onwards
5 Model Training	ILO 8.5A	Train a deep learning model for root segmentation	W2 onwards
6 Individual Root Segmentation	NA	Find individual roots	W3 onwards
7 Root System Architecture (RSA) Extraction	NA	Extract the root system architecture and measure the primary root length	W3 onwards
8 Kaggle Competition	ILO 8.5B	Compete on Kaggle to refine and optimize your root length prediction pipeline	W3 Mon - W8 Wed
9 Robotics Environment Interaction	ILO 8.6A	Send commands, retrieve observations of state, determine work envelope	W4 Mon
10 PID Controller	ILO 8.6B	Create a PID controller for the Opentrons OT-2	W4 Wed onwards
11 Reinforcement Learning Controller	ILO 8.6C	Train a reinforcement learning controller for the Opentrons OT-2	W4 Wed onwards
12 System Integration	ILO 8.6D	Integrate your CV pipelines with the robotic controllers	W6 onwards
13 Performance Benchmarking and System Evaluation	ILO 8.6D	Analyse the performance of the systems you have created	W6 onwards

Task	ILO	Description	Timeline
14 Client presentation	ILO 1.6, ILO 2.6, ILO 5.1, ILO 8	Present your combined computer vision and robotics solution to the client	W8 Fri

Plagiarism

⚠️ If plagiarism is detected in your deliverables, you will be reported to the Board of Examiners. ⚠️

- Always cite your sources.
- The deliverables must be your own work.
- You can use ChatGPT to get help with your code and writing.

GitHub repo

Here is the GitHub repository which contains code and documentation related to self-study and DataLab tasks for Year 2, Block B.

Datasets

Here are the available datasets:

1. **The Y2B_25 dataset.** 89 unlabeled images. You will label these images in Task 1. You will receive the labels during the block.
2. **The Y2B_23 and Y2B_24 datasets.** Datasets from the previous years with images and student labels.
3. **The Kaggle dataset.** Images without any labels or measurements, which will be your test set. You will predict the primary root lengths and submit them to Kaggle (Task 8).

You will annotate the Y2B_25 dataset to obtain segmentation masks (Task 1). Then, you will use these masks and images to develop a segmentation model with deep learning (Task 5). You can use the Y2B_23&24 datasets to improve your model's performance (optional but recommended). You will process the output of the segmentation model to measure primary root lengths (Task 8). The Kaggle dataset is provided as the test set of this competition. True primary root length measurements are hidden from you, but when you submit your predictions to Kaggle, a metric will be calculated and shown to you.

Task 1: Image Annotation and Review

Image annotation in computer vision involves marking specific regions or features within an image to provide additional information. This can include drawing bounding boxes, adding points, or segmenting areas, often coupled with labels to describe or categorize those regions.

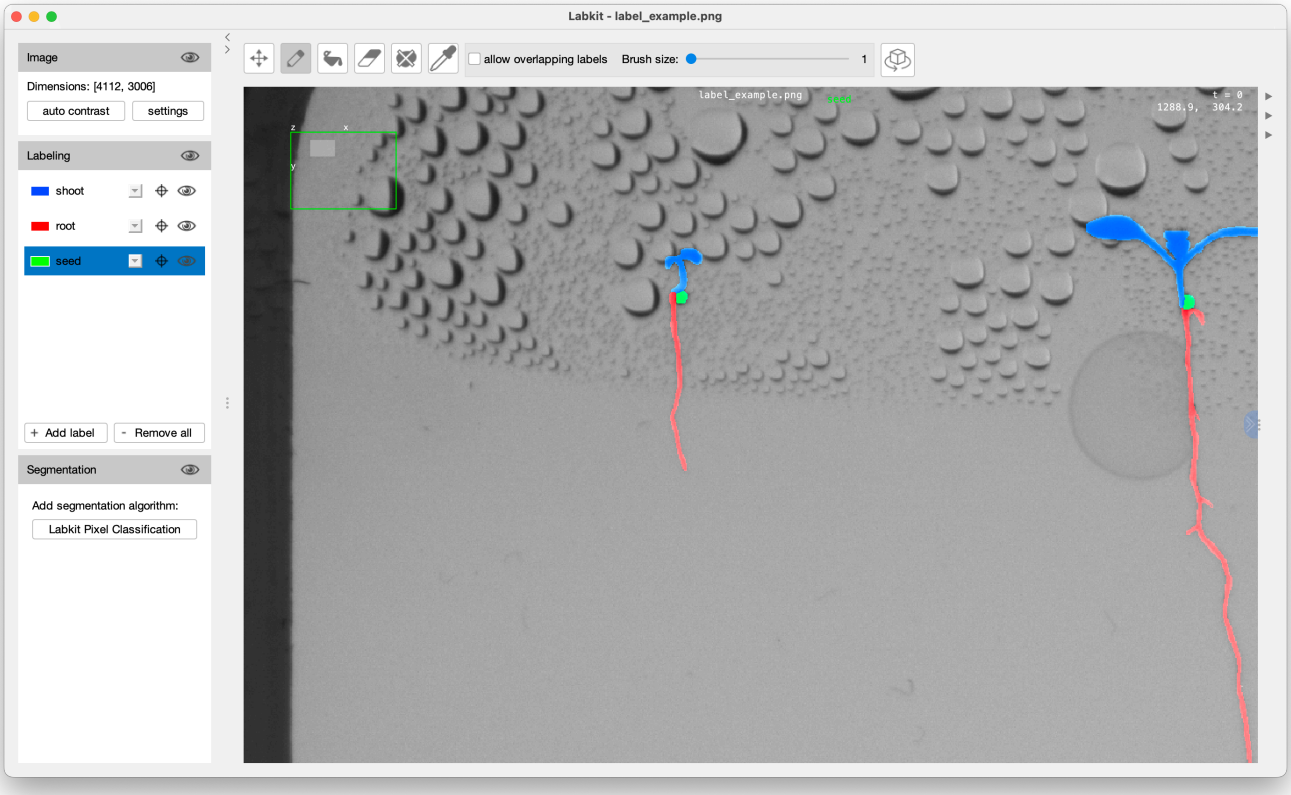
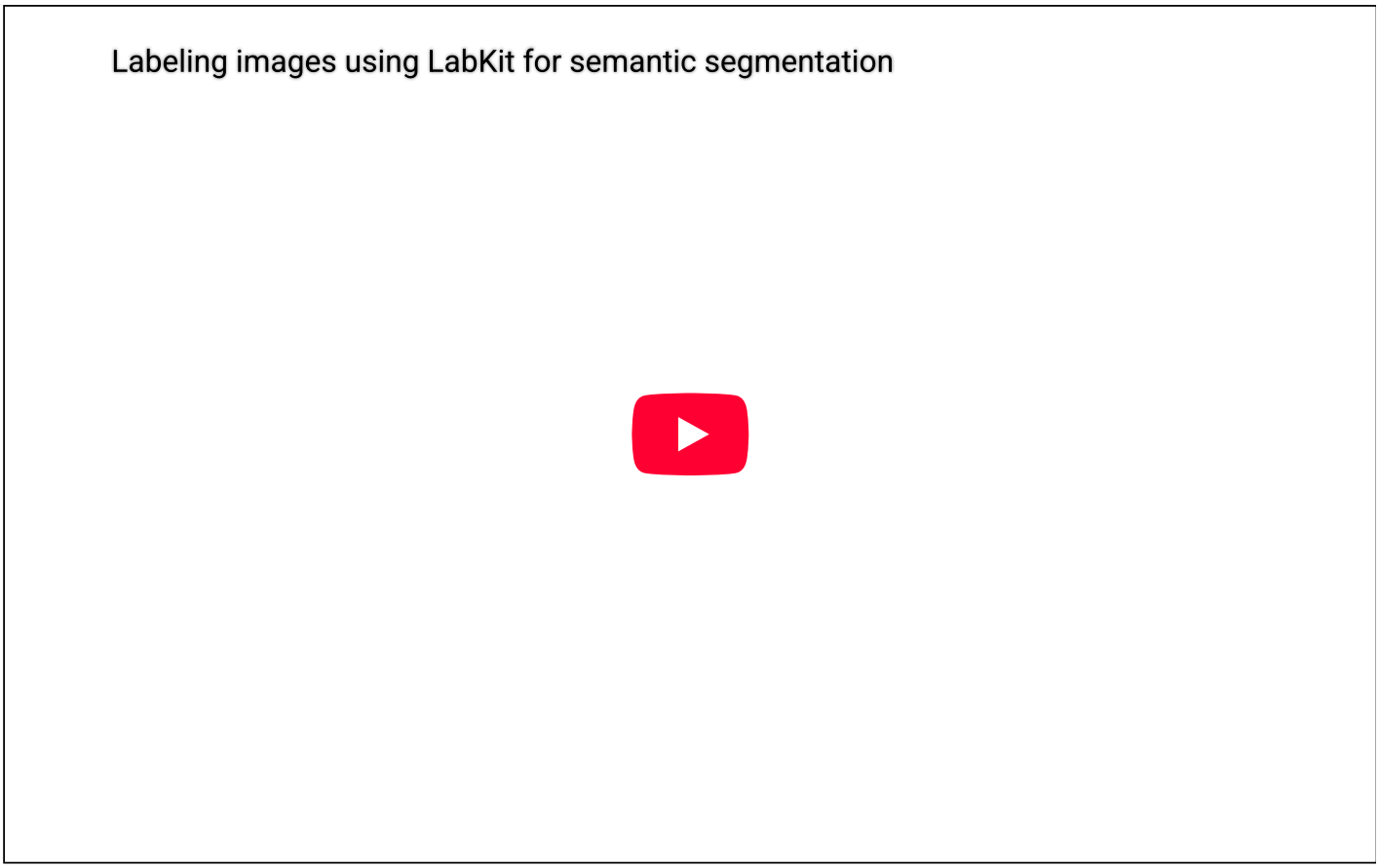


Figure 1.1: Example image annotation with LabKit. Make sure that the dimensions (top-left) are correct, brush size is 1 and allow overlapping labels option is unchecked.

The dataset of this project consists of black-and-white images without any labels or annotations. Finding root landmarks and measuring roots require pixel-level information. Therefore, we need this dataset to be labeled at the pixel level. Each student will annotate 1 image. You can find the images that are assigned to you by checking the image names, which contain mentor names and student IDs. (see the Y2B_25 dataset)

If the resulting segmentation masks satisfy the requirements, they will be added to the shared folder masks (see the Y2B_25 dataset). You need to label three classes: shoot, seed, and root. There are many tools for image annotation. Our suggestion is to use LabKit, an ImageJ plug-in. Figure 1.1 shows an example annotation with LabKit. Watch the following video to get started with LabKit.



Video 1.1: Labeling images using LabKit for semantic segmentation.

When you are opening LabKit, make sure to select the resolution that matches the image size.

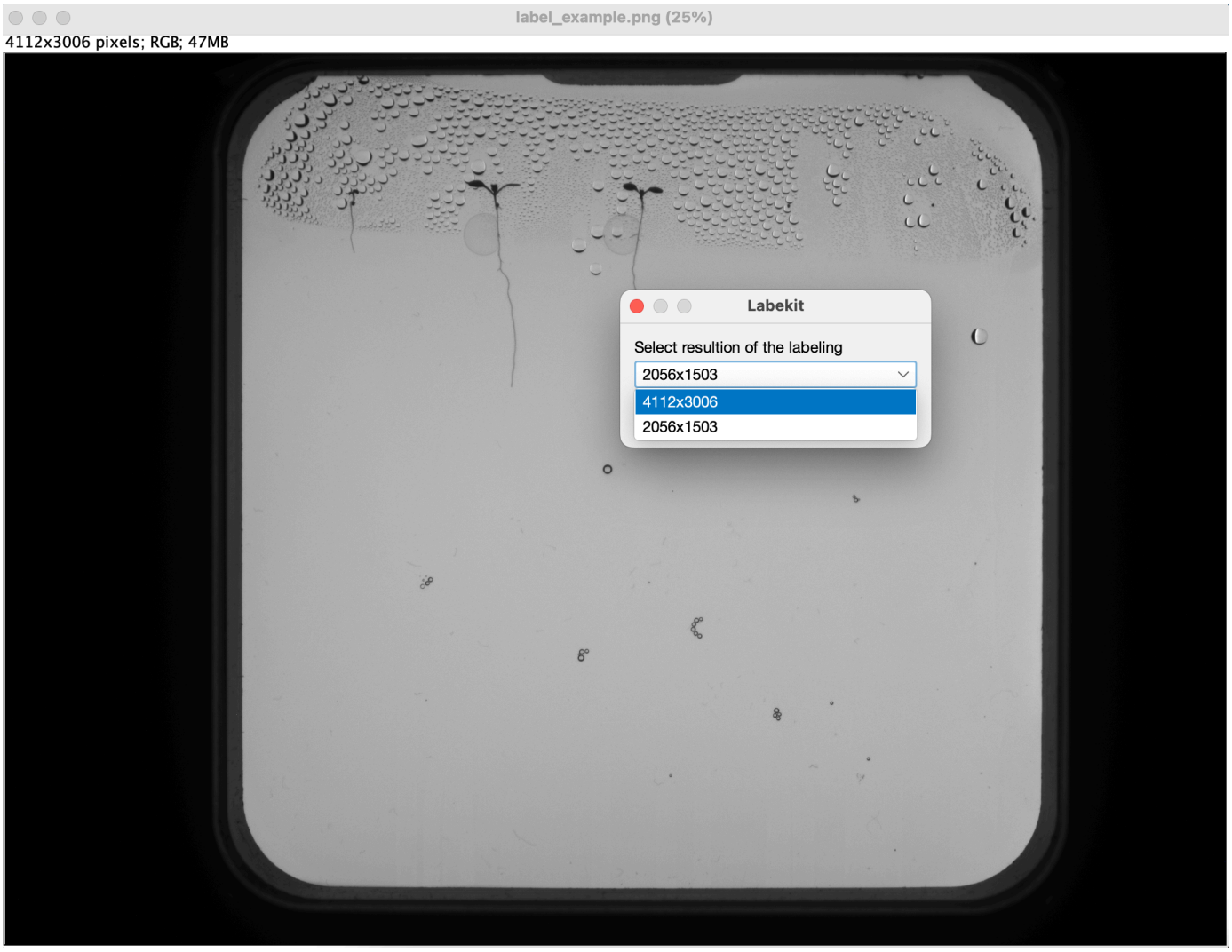


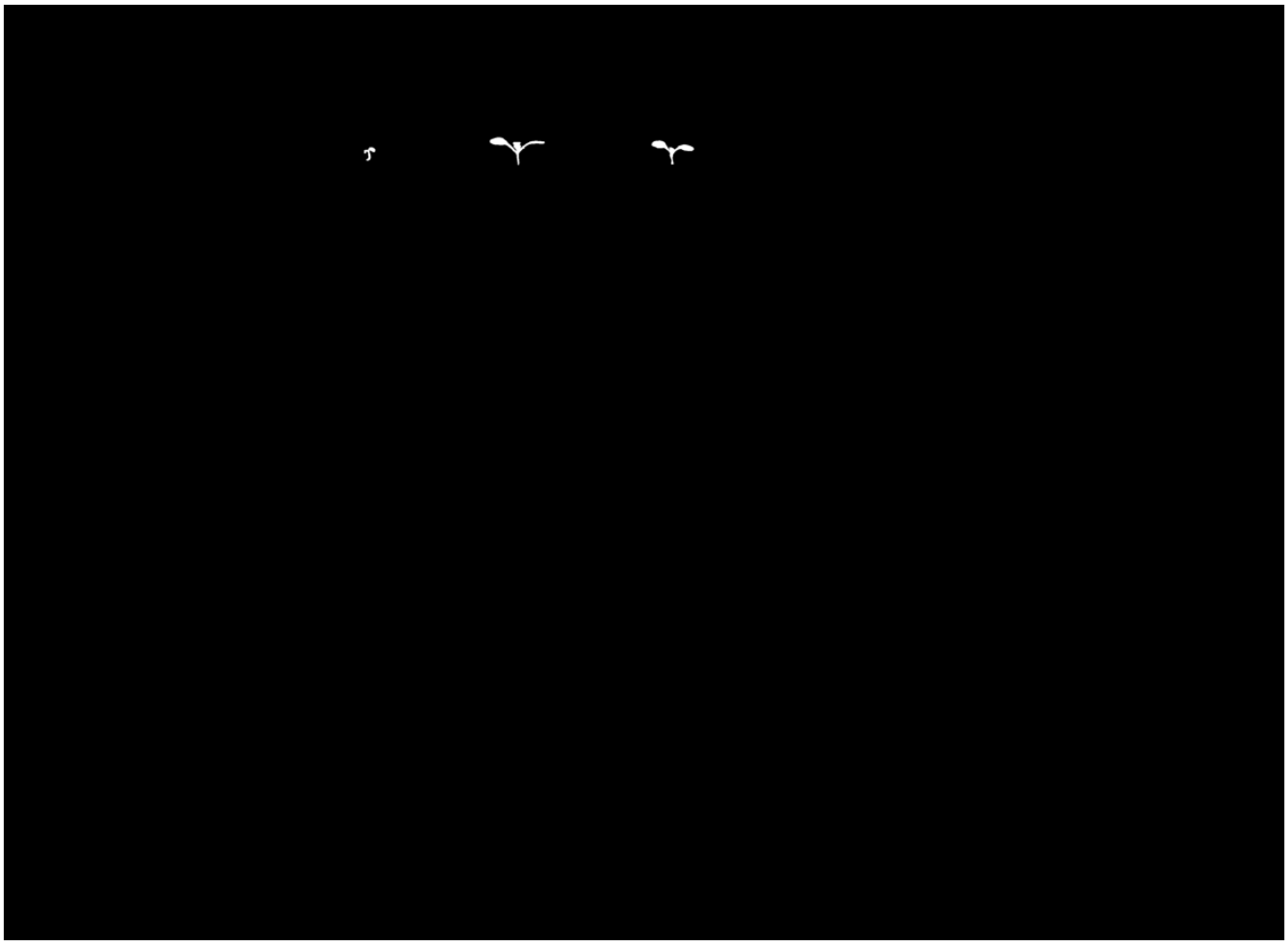
Figure 1.2: You should select the resolution that matches the resolution of the raw image to prevent scaling down.

As explained in the video, once you are done with the annotations, you can export them.

⚠️ **To avoid losing your labeling progress, do a quick dummy labeling first, then test saving and reloading the project.**

Figure 1.3 shows the segmentation masks exported from Figure 1.1.

Shoot Mask



Seed Mask

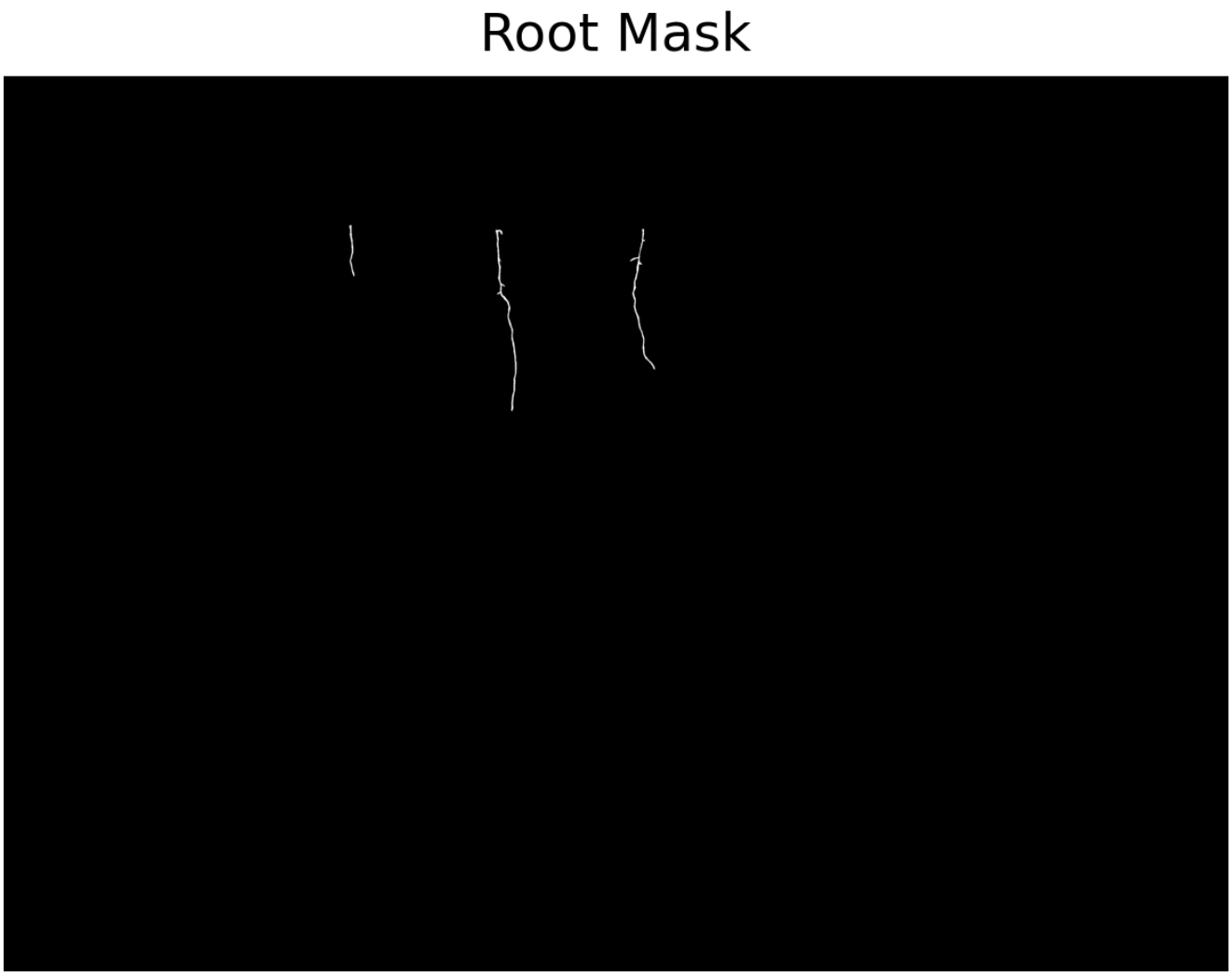


Figure 1.3: Segmentation masks.

Before labeling, take a look at labeling examples. The point where the shoot and root meet is called the junction. The labeling examples show you where the junction is. Make sure this boundary is labeled accurately, otherwise, your model may predict roots that are longer or shorter than they actually are.

Also watch the following short tutorial by our client Dr. Vinicius Lube. Note that they label primary and lateral roots as different classes, you do not have to do that, we only have a *root* class which consists of all root types.

Your labels will be reviewed by your peers and mentors according to the requirements explained below.

The Annotation and Review Process

This task follows a three-step process: initial labeling, peer review, and final submission. The goal is to use peer feedback to improve the quality of your annotation before it is graded by the mentors.

1. Initial Annotation: You label your assigned image. (Deadline W1 Friday 17:00)
2. Peer Review: You will review another student's initial annotation and provide constructive feedback. In parallel, another student will be reviewing your work. You should submit your review by filling in the peer review template. Requirements are here. You can find an example peer review from last year here. Note that last year each student labeled 5 images therefore the example contains 5 images. (Deadline W2 Friday 17:00)
3. Correction and Final Submission: You will receive the feedback from your peer. You must use this feedback to correct any mistakes in your annotation and submit the final version for grading. (Deadline W3 Friday 17:00)

Deliverables

Submit the following to the corresponding Brightspace Assignments:

- Task 1 - First Annotation
  - 3 masks, corresponding to the shoot, seed, and root classes for the image you are assigned to label.
  - One executed copy of the `task_1_requirements.ipynb` notebook to verify your labeled masks.
- Task 1 - Peer Review
  - The completed peer review template for the masks you were assigned to review.
- Task 1 - Final Annotation
  - 3 corrected masks, updated based on the peer feedback you received.
  - The final executed copy of the `task_1_requirements.ipynb` notebook for your corrected masks.

Client Requirements

The client requirements are explained in `task_1_requirements.ipynb`. Use this notebook to check your labels before submission and for reviewing your peers' masks. You can use any labeling tool as long as your deliverables meet the requirements. However, your mentors might be unable to provide support if you use a tool other than LabKit.

Your final annotation will be graded out of 3 points. We will use the following to grade your annotations. You will do the same to grade your peers.

Criteria	Grade
All masks pass Req 1–4, no label quality issues	3 points
All masks pass Req 1–4, a few label quality issues	2 points

Criteria	Grade
At least one mask passes Req 1–4, many label quality issues	1 point
No mask passes all of Req 1–4	0 point

Your peer reviews will be graded out of 2 points.

Criteria	Grade
Review submitted with the template, all major issues correctly identified and clearly described	2 points
Review submitted with the template, but only some issues detected or descriptions unclear	1 point
No review submitted using the provided template	0 point

ILO Mapping

ILO 6.4A (5 points), Prerequisite for ILO 8.5.

- Segmentation Masks (3 points)
- Peer review (2 points)
- Prerequisite for ILO 8.5.
  - To satisfy this prerequisite, you must achieve a score of at least 2 out of 5.

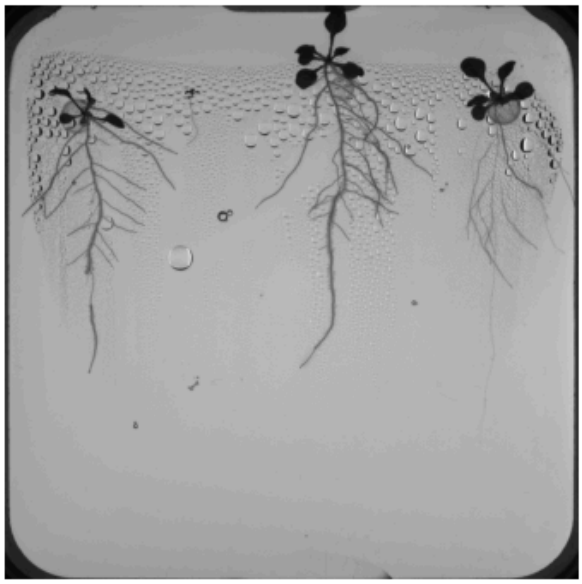
Evidencing

No evidence link is required. In your learning log section C, under ILO 6.4A, briefly describe your assignment results. For example: “I delivered masks with minor quality issues (2/3) and provided an accurate peer review (2/2)”.

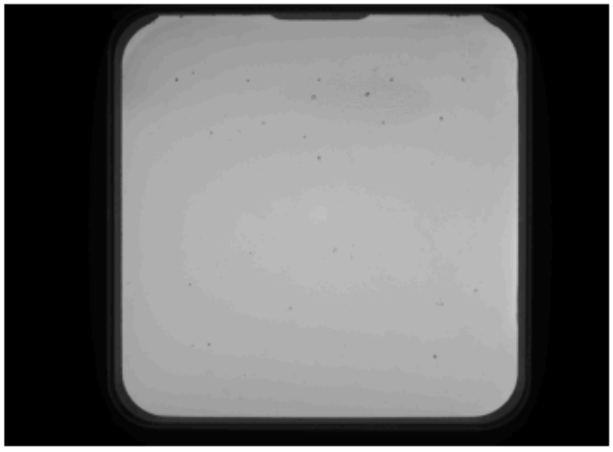
Task 2: Petri Dish Detection and Extraction

In Computer Vision, Region of Interest (ROI) extraction involves identifying and isolating specific parts of an image deemed important for further analysis. This process allows focused processing on these selected areas, often enhancing efficiency and accuracy.

(2736, 2736)



(2737, 2737)



(2774, 2774)

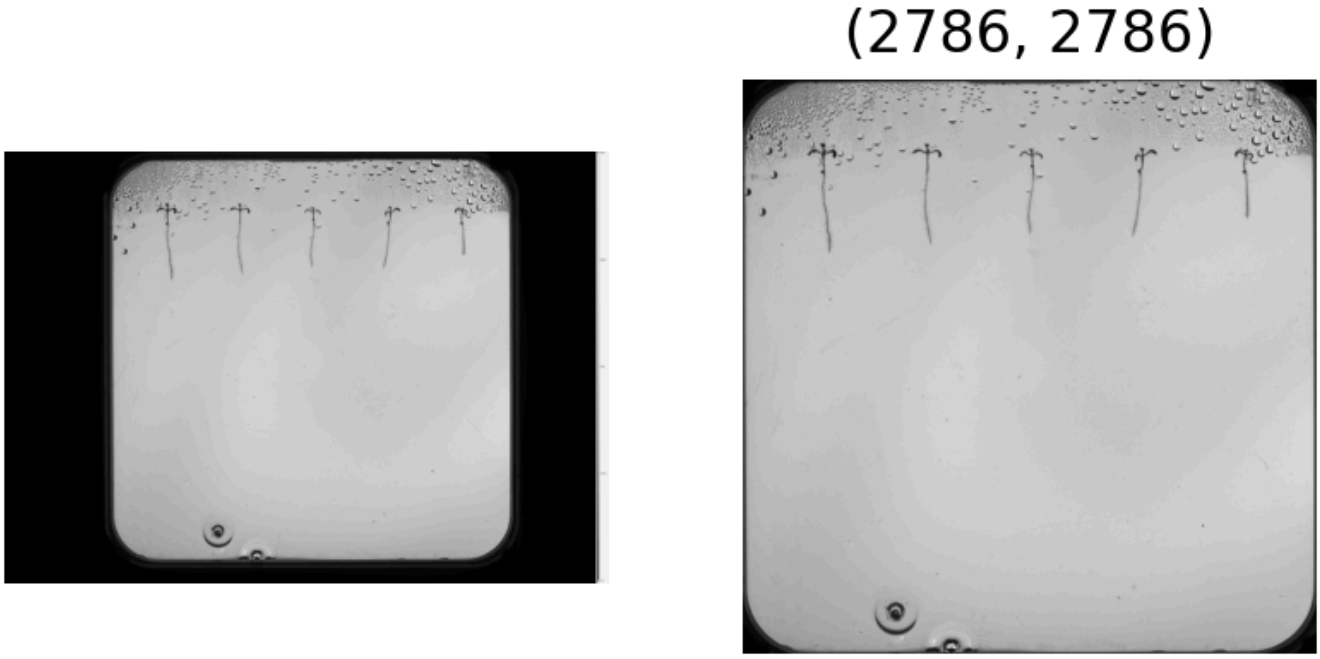


Figure 2.1: Petri dish extraction from Y2B_23 and Y2B_24

In this project, the ROI is the Petri Dish. Your task is to write a Python code using traditional computer vision methods that crop the Petri dish from any raw image from the datasets Y2B_23, Y2B_24 and Y2B_25. Note that these datasets are slightly different. For example, Y2B_23 images contain a white legend on the right edge, while Y2B_24 images do not. Also, Petri Dishes are different. In the following tasks, you will use the Petri Dish extraction code so that you don't have to process outside the Petri Dish where no plant is present. This will save you time and compute. But more importantly, the model will learn better. You can achieve this task with deep learning, but that would be overkill; therefore, using traditional CV methods is a requirement.

Please assume that the Petri dish is a square, ignoring corner curvature, with its edges parallel to the image borders. However, its location can change from one image to the other. The cropped image must be a square.

Deliverables

- The code (`task_2.ipynb`), with comments/docstrings explaining each step. The notebook should display the output of the code on one random image from Y2B_23, one random images from Y2B_24 and two random images from Y2B_25 (Figure 2.1).

Client Requirements

1. Your code should rely on traditional computer vision methods. In other words, do not use an ML/DL model.
2. Detected Petri dish edges must be within ± 30 pixels of the actual Petri dish edges.
3. The code should be general enough to work on all the dataset images (Y2B_23, Y2B_24, Y2B_25).
4. The output of the code must be a square.
5. An image similar to Figure 2.1 is shown in `task_2.ipynb`.

ILO Mapping

ILO 6.4B (3 points), no partial points. If you fail to meet the requirements or if you have evidencing issues, you will receive 0 from ILO 6.4B.

Evidencing

In your learning log section C, under ILO 6.4B, provide a link to `task_2.ipynb` and describe how you have solved this task in a few sentences.

Task 3: Plant Instance Segmentation

Instance segmentation is the task of classifying and distinguishing each individual object (i.e. instance) in an image at the pixel level. Unlike semantic segmentation, which groups all objects of the same class together, instance segmentation differentiates each distinct object, even if they belong to the same class.

Figure 3.1: Segmenting individual plants.

For this task, you will need to do instance segmentation for five plants. Do not segment plant organs, simply find individual plants. Use only traditional computer vision techniques. Your code should at least work on two images, `task_3_image_1.png` and `task_3_image_2.png`, which you can find here.

Deliverables

- The code (`task_3.ipynb`), with comments/docstrings explaining each step. The notebook should display the output of the code on `task_3_image_1.png` and `task_3_image_2.png`.

Client Requirements

1. Your code should only rely on traditional computer vision methods. In other words, do not use an ML/DL model.
2. The quality of the segmentation will be qualitatively assessed by comparing it to Figure 3.1
3. The code should work for at least `task_3_image_1.png` and `task_3_image_2.png`.
4. An output similar to Figure 3.1 is shown in `task_3.ipynb` for the images `task_3_image_1.png` and `task_3_image_2.png`.

ILO Mapping

ILO 6.4C (2 points), no partial points. If you fail to meet the requirements or if you have evidencing issues, you will receive 0 from ILO 6.4C.

Evidencing

In your learning log section C, under ILO 6.4C, provide a link to `task_3.ipynb` and describe how you have solved this task in a few sentences.

Task 4: Dataset Preparation for Model Training

In Task 5, you will train a deep learning model to detect root pixels. To prepare for this, you need to organize and preprocess your dataset as demonstrated in the Week 2 self-study materials. This involves arranging the dataset files into a specific directory structure and generating image patches (Figure 4.1).

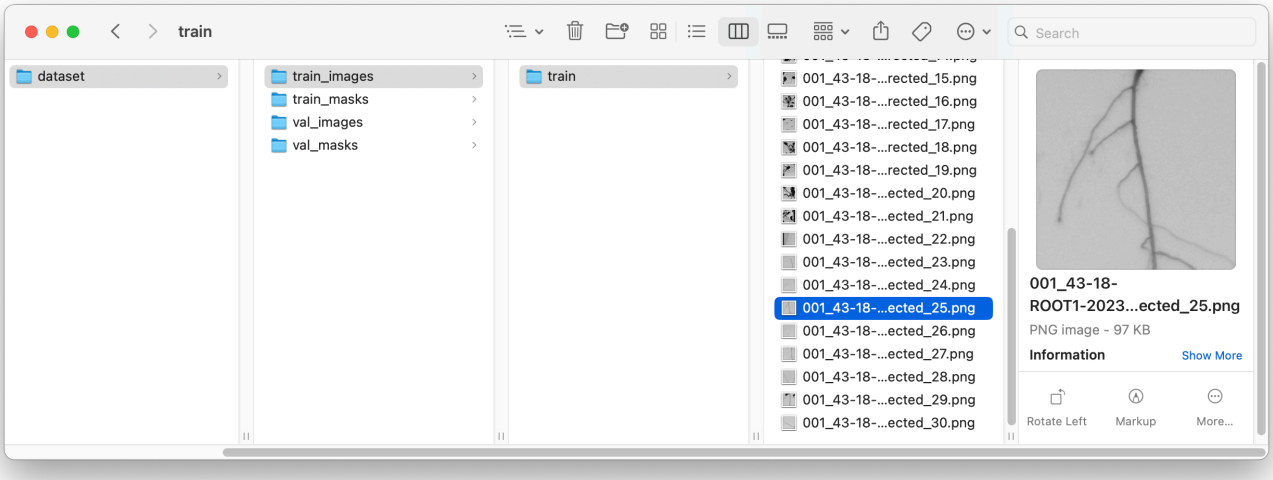


Figure 4.1: Expected directory structure.

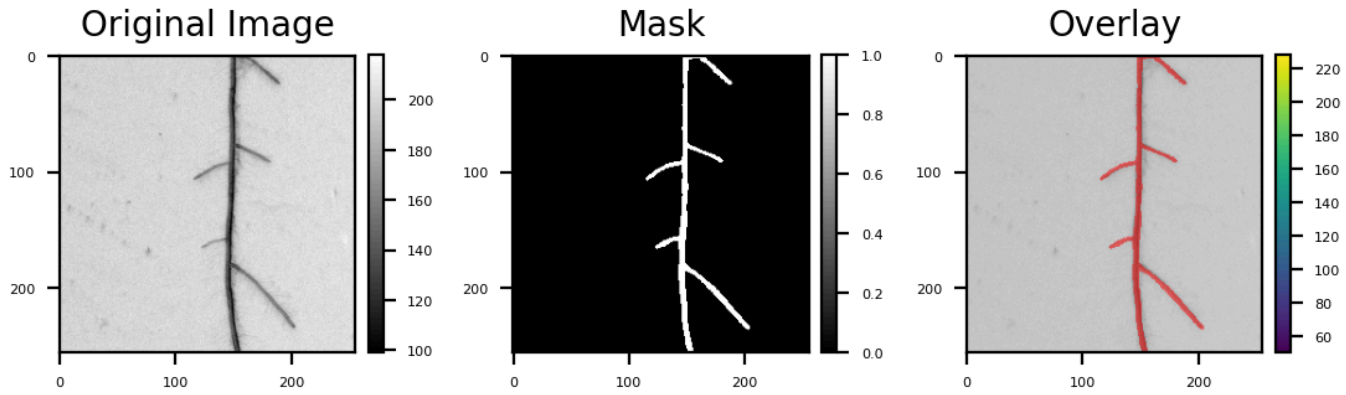


Figure 4.2: A random image patch, the corresponding binary mask highlighting root pixels, and an overlay of the mask on the image patch. This setup allows for easy verification of alignment and accuracy. Check that the patch size and pixel values are correct by examining the axes and color legends.

It is crucial to complete this dataset preparation task correctly before proceeding to Task 5. Along with writing code to process the dataset, you should also implement code to verify your work. For instance, you can write a code snippet to generate Figure 4.2, which will help you check the alignment and correctness of your dataset preprocessing. Make sure that every time you run this snippet it gives you a random patch. Check at least 20-30 images to make sure everything works.

This task is not graded directly, but it is essential for the successful completion of Task 5, which is graded. No deliverables are required for this task, as it serves as a preparation step for the next task.

Deliverables

None

Client Requirements

None

ILO Mapping

None

Evidencing

None

Task 5: Model Training

Until this task, you have been busy preparing a dataset to train a segmentation model. In this task, you'll use this dataset to develop a segmentation model.

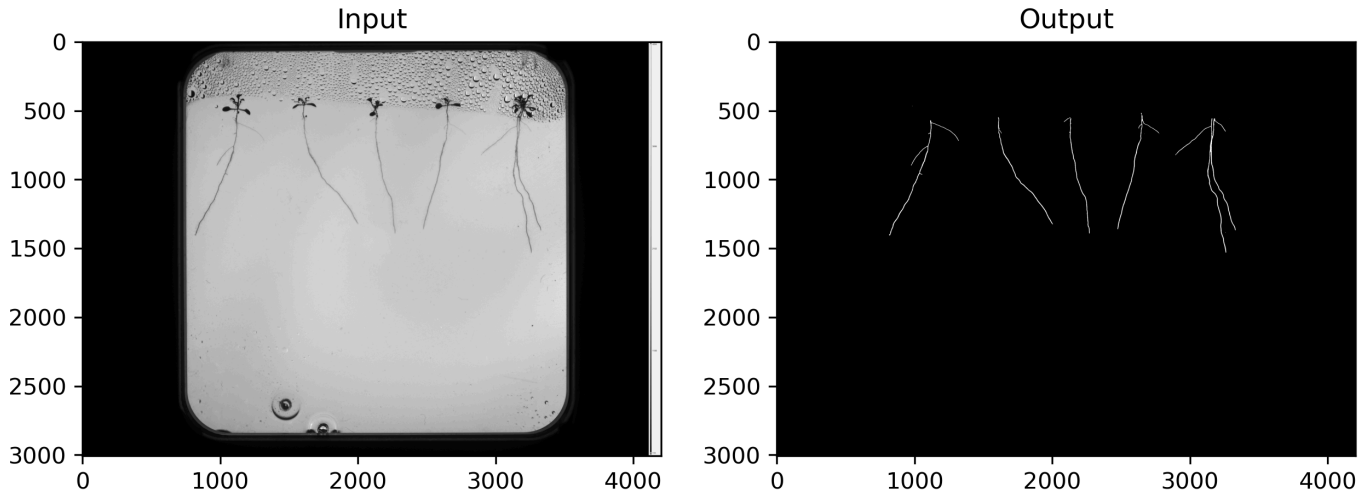


Figure 5.1: Prediction results of a U-Net model trained on Y2B_23. The left image displays the original input, while the right image shows the binary mask output. The input image is from Y2B_23.

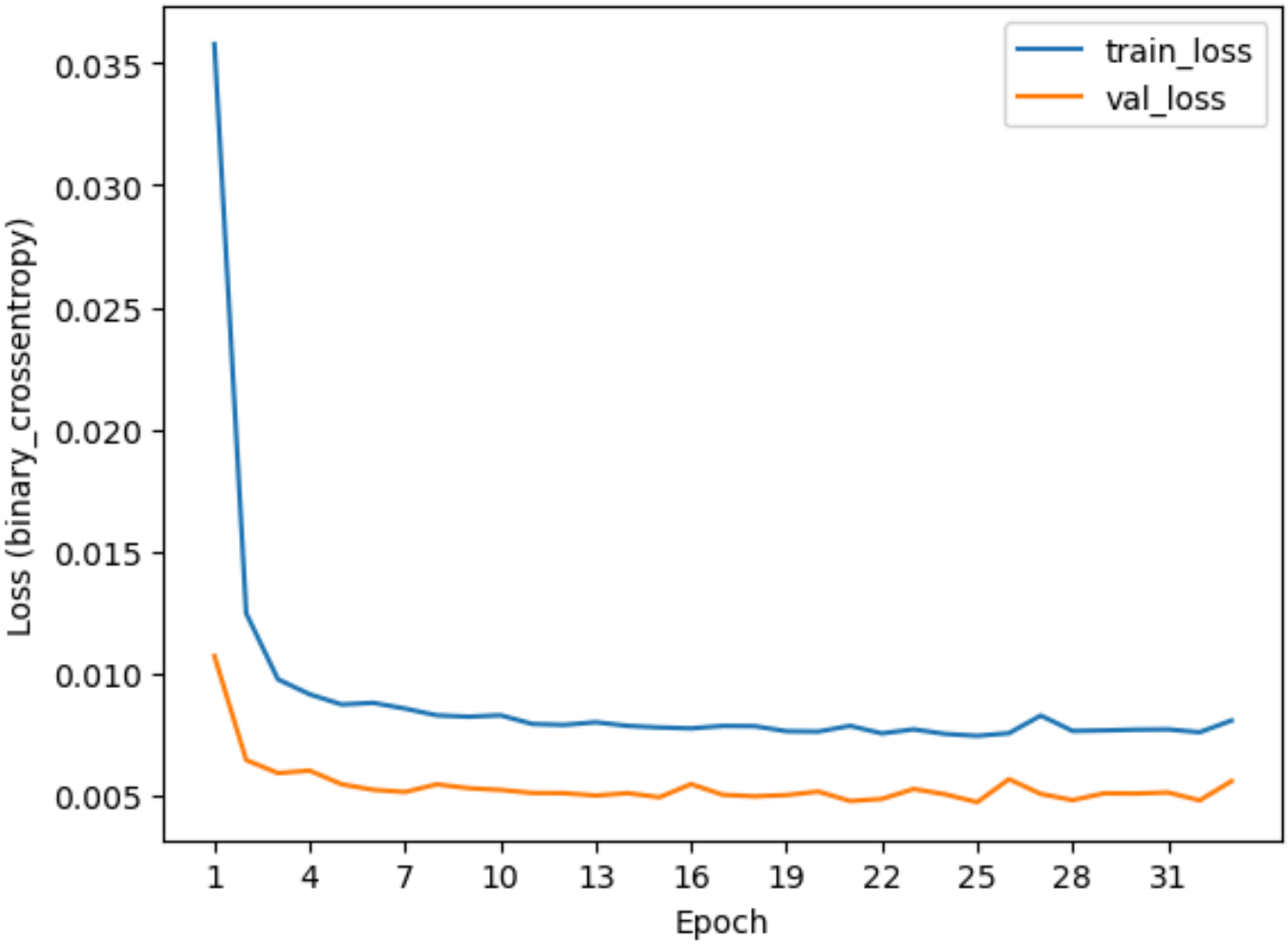


Figure 5.2: Training and validation binary cross-entropy loss for the U-Net model.

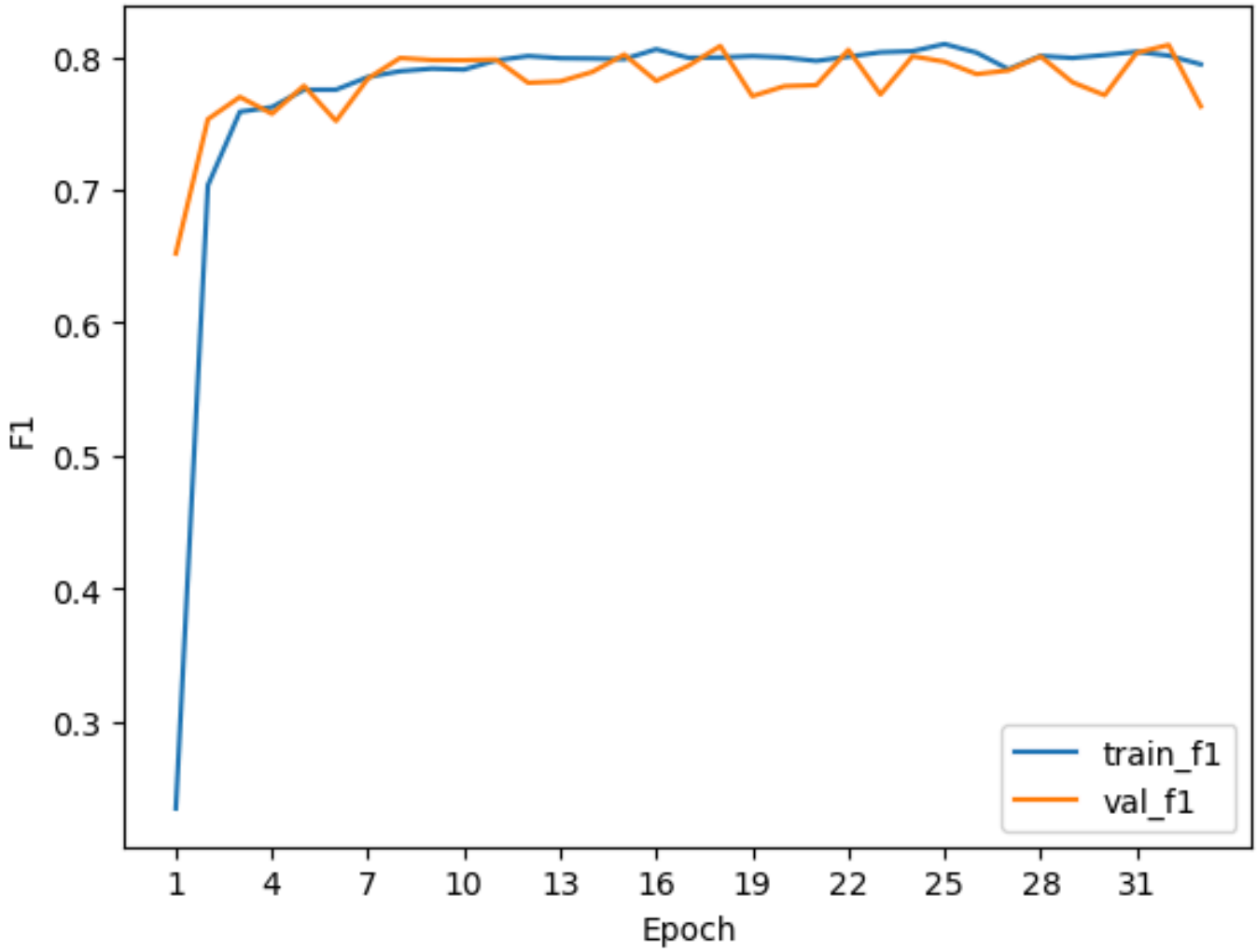


Figure 5.3: Training and validation F1 scores for the U-Net model.

You are expected to train a model that combines datasets Y2B_23, Y2B_24 and Y2B_25. Consequently, your results may differ from Figures 5.1 to 5.3.

Additionally, in this task, you are required to write inference code that accepts a raw image as input and outputs the predicted mask.

Deliverables

Please follow these templates for your deliverables.

- Training code: `task_5_training.ipynb`
- Inference code: `task_5_inference.ipynb`
- Trained model file: `studentname_studentnumber_unet_model_patchsizepx.h5`
- Predicted mask for `task5_test_image.png`, generated by the model you delivered. Save the output as: `task5_test_image_prediction_studentname_studentnumber.png`

Client Requirements

1. Validate your model using images labeled "val_..." in Y2B_25. Do not use a different validation set.
2. You may use any data for training except the validation set defined above.
3. The `val_f1` is 0.8.
4. The `val_f1` must be >=0.5.

5 Follow all requirements specified in the templates.

ILO Mapping

ILO 8.5A (10 points)

Validation F1 Score	Outcome
`val_f1` ≥ 0.8	10
0.5 ≤ `val_f1` < 0.8	Points based on class performance
`val_f1` < 0.5	0

Note: Grading is based on both `val_f1` and the quality and completeness of deliverables.

Evidencing

In your learning log, Section C, under ILO 8.5A, include links to all the required deliverables. Missing any deliverables will result in a score of zero for this ILO. Additionally, provide a brief explanation of your work under ILO 8.5A. For example:

"I trained a U-Net model for root segmentation, achieving an F1 score of 0.8021 on the validation set. I also developed inference code capable of processing raw images from Hades and generating the corresponding predicted masks."

Task 6: Individual Root Segmentation

In Task 3, you applied instance segmentation to the plants using traditional CV. However, obtaining roots with this approach would be very difficult. In this task, you will do instance segmentation at the root level. Thanks to deep learning, you already have a root mask from Task 5. Apply instance segmentation on this image to find individual roots.

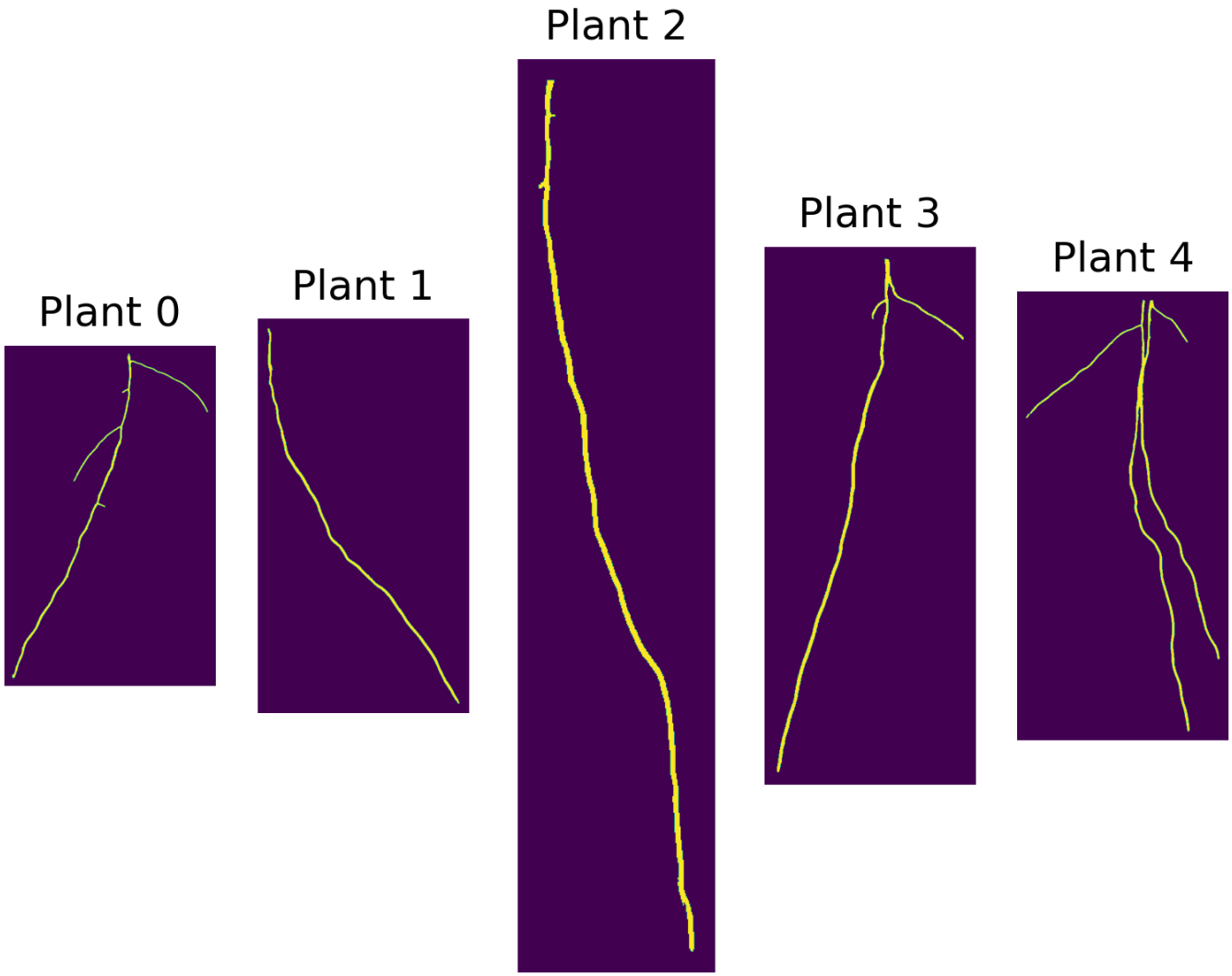


Figure 6.1: Segmented root structures for individual plants extracted from the full mask shown in Figure 5.1. Each subplot displays an isolated root mask for a different plant. These segmented masks will be used in Task 7 to extract individual root system architectures.

No deliverables are required for this task, as it serves as a preparation step for the next task.

Deliverables

None

Client Requirements

None

ILO Mapping

None

Evidencing

None

Task 7: Root System Architecture (RSA) Extraction

A root mask distinguishes root pixels (1) from non-root pixels (0), but it does not convey any information about the relationships between root branches. In this task, you will extract the root system architecture (RSA) for each plant from the root mask. RSA allows you to classify root types (primary root, secondary root etc.) and to find root tips.

	skeleton-id	node-id-src	node-id-dst	branch-distance	branch-type	mean-pixel-value	stdev-pixel-value	image-coord-src-0	image-coord-src-1	image-coord-dst-0	image-coord-dst-1	coord-src-0	coord-src-1	coord-dst-0	coord-dst-1	euclidean-distance
0	0	0	12	8.414214	1	1.0	0.0	20	319	28	320	20	319	28	320	8.062258
1	0	5	12	4.828427	1	1.0	0.0	25	317	28	320	25	317	28	320	4.242641
2	0	12	29	13.242641	2	1.0	0.0	28	320	40	323	28	320	40	323	12.369317
3	0	29	383	263.462987	1	1.0	0.0	40	323	170	523	40	323	170	523	238.537209
4	0	29	258	75.899495	2	1.0	0.0	40	323	113	320	40	323	113	320	73.061618
5	0	258	275	16.071068	1	1.0	0.0	113	320	118	306	113	320	118	306	14.866069
6	0	258	426	105.112698	2	1.0	0.0	113	320	209	300	113	320	209	300	98.061205
7	0	426	712	196.906638	1	1.0	0.0	209	300	347	179	209	300	347	179	183.534738
8	0	426	774	223.681241	2	1.0	0.0	209	300	407	240	209	300	407	240	206.891276
9	0	774	1238	534.055916	2	1.0	0.0	407	240	850	23	407	240	850	23	493.293016
10	0	774	799	20.899495	1	1.0	0.0	407	240	414	258	407	240	414	258	19.313208
11	0	1232	1238	3.828427	1	1.0	0.0	847	21	850	23	847	21	850	23	3.605551
12	0	1238	1241	2.414214	1	1.0	0.0	850	23	852	22	850	23	852	22	2.236068

Figure 7.1: The RSA of Plant 0, displaying the root system with 13 identified branches, along with their start and end points, and additional structural details.

The final step in this task is to measure the primary root length. You will also need to locate the primary root tip coordinates, as these coordinates are essential for downstream robotics tasks that involve inoculating plants at their root tips. Root tip locations are not needed for Tasks 7 and 8.

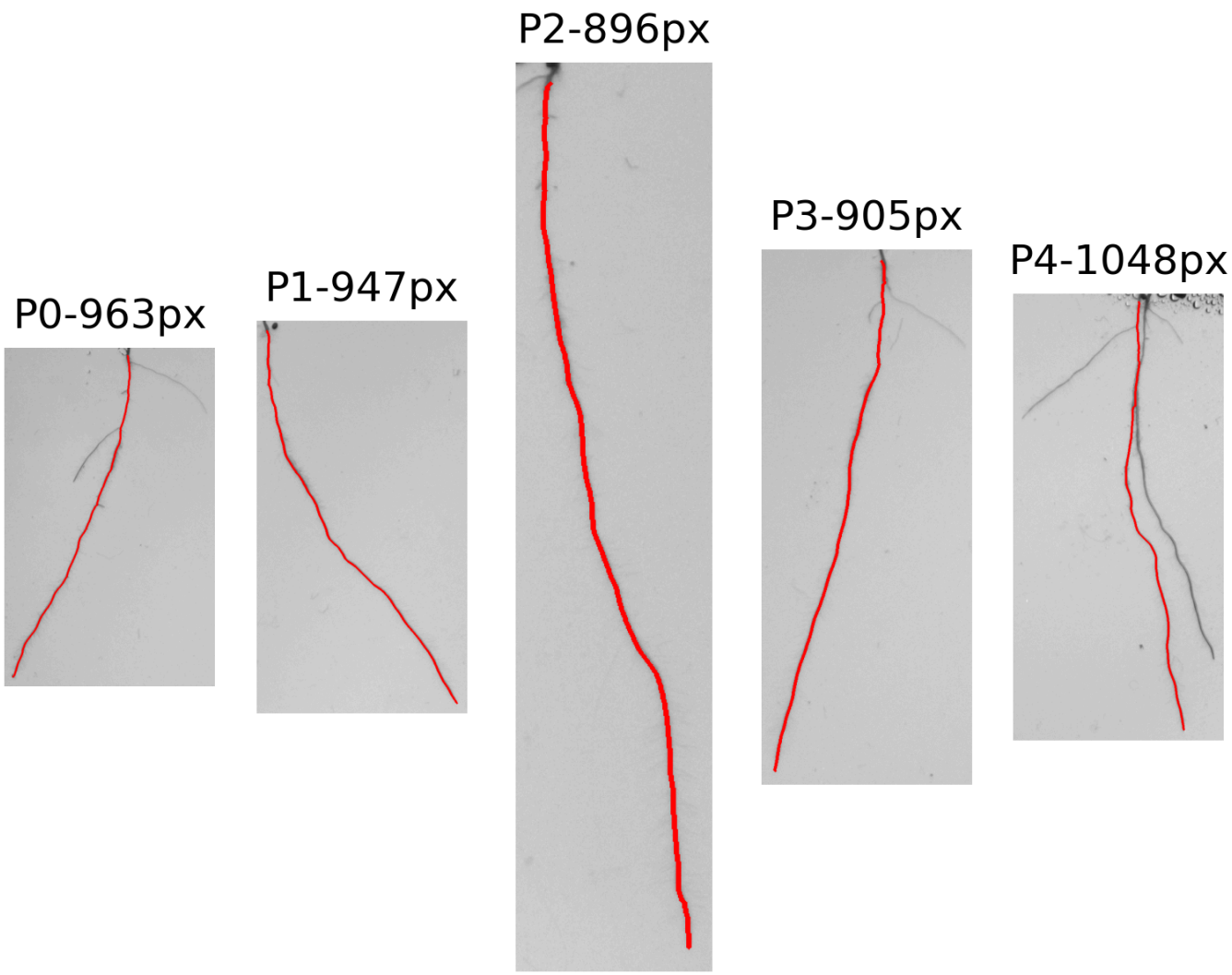


Figure 7.2: The red line indicates the predicted primary roots, with the predicted root length displayed above each image.

No deliverables are required for this task, as it serves as a preparation step for the next task.

Deliverables

None

Client Requirements

None

ILO Mapping

None

Evidencing

None

Task 8: Kaggle Competition

Start: W3 Wednesday 14:00

End: W8 Wednesday 14:00

For this task, you will integrate the skills and knowledge acquired in Tasks 1 through 7 to construct a comprehensive computer vision pipeline. The pipeline should take a directory containing raw images from Hades as input and output the primary root length for each plant in every image. The pipeline should not contain any training code; it should include only preprocessing, inference, and post-processing steps.

Note that Tasks 1 to 7 are designed to give you a starting point, but you should go beyond these tasks. For instance, consider developing additional segmentation models for other plant organs labeled in Task 1. These models could assist in locating individual plants and isolating primary roots more accurately. Take advantage of this flexibility to push the boundaries of your solution.

To assess the performance of your solution, we turned this task into a Kaggle competition. The competition will run from W3 Mon to W8 Wed.

1. Create a Kaggle account with your buas.nl student email.
2. Join the competition using the following link.
3. Predict the primary root lengths of the plants in the Kaggle dataset. (Teams>Y2B Team>Files>Datasets)
4. Submit your predictions in the format explained in the competition.

On Kaggle, there are two leaderboards: the **public leaderboard**, which shows scores based on a portion of the test data and updates during the competition, and the **private leaderboard**, which is hidden until the end and determines the final rankings. This approach helps reduce overfitting to the public data and encourages models that generalize well.

Throughout the competition, you will make multiple submissions, experimenting with different models, data, and processing steps, and you'll see each submission's results on the **public leaderboard**. Just before the

competition closes, you must select two of your submissions to be evaluated on the private leaderboard, so it's essential to keep track of your submission pipelines. Once the final results are announced, you'll see which of your two chosen submissions performed best on the private data. For the best of the two selected submissions, you'll need to deliver your code (Task 8) and present its details (Task 14).

Deliverables

- Well-documented code for your best-performing pipeline based on the private leaderboard. You may submit your code in one of the following formats:
  - A single notebook: `task_8.ipynb`
  - Multiple notebooks, labeled sequentially: `001_task_8.ipynb`, `002_task_8.ipynb` ...
  - A single notebook with helper functions in a separate script: `task_8.ipynb` and `task_8_helpers.py`
- The root segmentation model used in the pipeline. This can be the same model developed in Task 5 or an improved version.
- If your pipeline includes other models, such as a shoot segmentation model for better plant localization, you must deliver these as well. For example, if you developed a shoot segmentation model to enhance plant detection, ensure it is included as part of your submission.
- Presentation of your pipeline at the final block presentation.
- Screenshot of your private leaderboard position and score.

Client Requirements

1. The desired sMAPE is less than 10% (Private LB score on the Kaggle competition).
2. The sMAPE must not exceed 45% (Private LB score on the Kaggle competition).
3. Your solution must utilize a deep learning model for root segmentation, followed by Root System Architecture (RSA) extraction.
4. The pipeline should not contain any training code; it should include only preprocessing, inference, and post-processing steps.
5. *Pipeline Input:* A directory containing any number of raw images. *Pipeline Output:* The primary root length for every plant in each image.

ILO Mapping

ILO 8.5B (20 points)

Private sMAPE (%)	Outcome
sMAPE < 10%	20
10% ≤ sMAPE < 45%	Points based on private leaderboard
45% ≤ sMAPE	0

Note: Grading will consider both your private leaderboard sMAPE and the quality and completeness of your deliverables. For instance, achieving an excellent leaderboard score but failing to clearly explain your pipeline

during the presentation may result in a deduction of points.

Evidencing

In your learning log, Section C, under ILO 8.5B, include links to all the required deliverables. Missing any deliverables will result in a score of zero for this ILO. Additionally, provide a brief explanation of your work under ILO 8.5B. For example:

"I developed a computer vision pipeline capable of processing any number of raw images from Hades and generating the primary root lengths for each plant. My predictions were submitted to the Kaggle competition, achieving a 13% sMAPE on the private leaderboard and securing 10th place in the competition." This is followed by the Kaggle private LB screenshot and links to other deliverables.

Robotics Tasks

You can complete Tasks 9, 10, and 11 without the computer vision pipeline. Don't wait until your CV pipeline is ready, start these tasks in Week 4, even if the pipeline isn't finished yet. Only from Task 12 onward, you'll need the CV pipeline to continue.

For the Robotics Tasks, we want you to **document your approach clearly using README files.** Create the following folders in your repository:

- `task09_robotics_environment`
- `task10_pid_controller`
- `task11_rl_controller`
- `task12_13_integration_benchmarking`

In each folder, create a `README.md` file. Document your approach, implementation steps, and key results in the corresponding task's `README.md` file. Include relevant code snippets, visualizations, and short explanations linking your work to the deliverables. Ensure that all specific requirements defined in each task are documented in the corresponding `README.md` file. For example, the work envelope should be included in `task09_robotics_environment/README.md`.

Task 9: Robotics Environment Interaction

In this task you will demonstrate your ability to interact with a simulation environment and understand fundamental robotics concepts. You will interface with the Opentrons OT-2 simulation to command the robot and receive observations about its state. The goal is to explore the robot's capabilities by understanding and manipulating key parameters such as position, velocity, and distance to target locations. You will also determine the work envelope (operational boundaries) of the pipette tip by moving it to all corners of the workspace and recording coordinates.

For more details on how to set up and interface with the simulation environment click here.

Figure 9.1: Opentrons OT-2

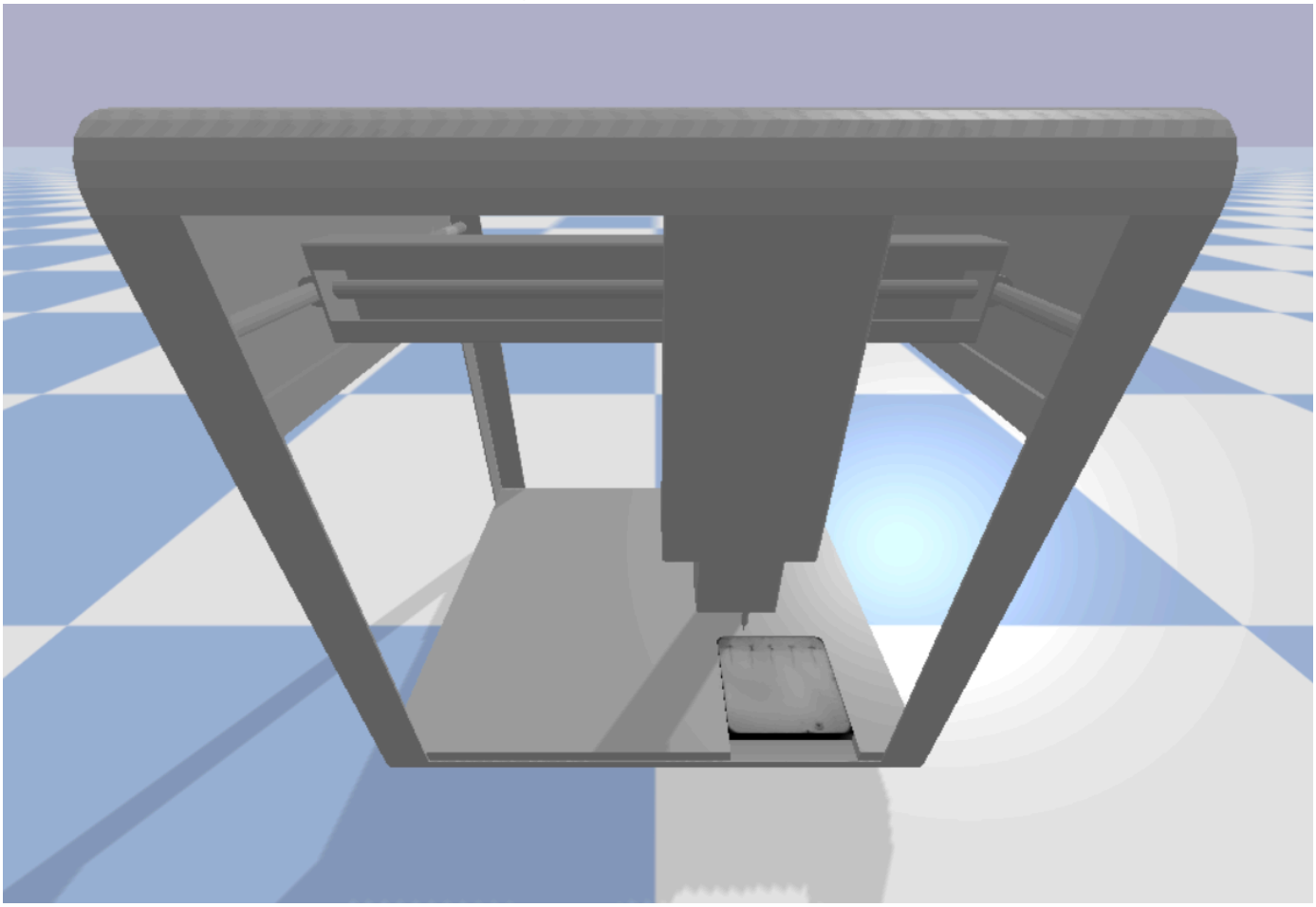


Figure 9.2: The robotic simulation environment.

Deliverables

- `task09_robotics_environment/README.md` which includes determined work envelope coordinates, environment setup instructions and dependencies, and key findings on robot behavior/limitations.
- Well documented code that interfaces with the simulation, sends commands, and processes observations

- GIF(s) demonstrating the robot responding to commands with printed observations visible

ILO Mapping

ILO 8.6A

Evidencing

In your learning log section C, under ILO 8.6A, provide:

- Link to `task09_robotics_environment/README.md`
- Link to the code for interfacing with the simulation, sending commands, and processing observations
- GIF of the pipette moving to all 8 corners

Task 10: PID Controller

In this task you will design and implement a PID (Proportional-Integral-Derivative) controller for the Opentrons OT-2. The goal is to enable precise movement of the pipette tip to any given position within the work envelope. You will need to tune the PID controller parameters (Kp, Ki, Kd) to achieve optimal performance.

Hint: You will need three PID controllers, one for each axis (X, Y, Z).

Deliverables

- `task10_pid_controller/README.md` with comprehensive documentation including implementation steps, tuning strategies, libraries used, performance metrics and error analysis, and tuning results with final gains.
- Well documented working code for the PID controller implementation
- Well documented code for the PID controller testing and tuning process
- GIF(s) of robot moving to target coordinates controlled by the PID controller
- Presentation slides showing tuning results and final gains

Client Requirements

Positioning Error	Points
< 0.001 m (1 mm)	7
0.001 m ≤ error < 0.005 m	5
0.005 m ≤ error < 0.01 m (10 mm)	3
0.01 m (10 mm) ≤ error	0

ILO Mapping

ILO 8.6B

Evidencing

In your learning log section C, under ILO 8.6B, provide:

- Link to `task10_pid_controller/README.md`

- Link to the PID controller implementation code
- Link to the PID controller tuning & testing code
- GIF(s) of robot moving to target coordinates controlled by the PID controller
- Link to presentation slides showing tuning results and final gains

Task 11: Reinforcement Learning Controller

In this task you will design and implement a reinforcement learning (RL) controller using Stable Baselines 3 to control the Opentrons OT-2. First, you will create a Gymnasium-compatible wrapper for the simulation environment, defining action spaces, observation spaces, reward functions, and termination conditions. Then you will train an RL agent to move the pipette tip to any position within the work envelope.

You will conduct a hyperparameter search in groups of 5 students to find the best performing algorithm, hyperparameters, reward function, and termination conditions. Each group member will test assigned hyperparameter sets, with a leaderboard tracking speed and accuracy across groups.

Hints: Use Weights & Biases to track experiments. Training typically requires 1-2 days on the GPU server using the ClearML job queue.

Deliverables

- `task11_rl_controller/README.md` including implementation steps and design choices, tuning strategies and libraries used, performance metrics with error analysis and comparison to the PID controller, and documentation of best hyperparameters and saved model weights
- Well documented code for the Gymnasium wrapper (`ot2_gym_wrapper.py`) with all required properties and methods
- Simple test script (`test_wrapper.py`) running the environment for 1000 steps with random actions
- Well-documented RL training code (individual submission) showing the training loop, logging, and model saving
- Well-documented RL testing code (individual submission) demonstrating how to load the trained model and evaluate performance
- Best individual and group model weights
- GIF(s) showing the trained RL agent successfully reaching target positions
- Visualizations comparing RL controller performance with traditional control methods across defined metrics
- Presentation slides summarizing results, comparison plots, and hyperparameter search process and results.

Client Requirements

Positioning Error	Points
< 0.001 m (1 mm)	8
0.001 m ≤ error < 0.005 m	6
0.005 m ≤ error < 0.01 m (10 mm)	4
0.01 m (10 mm) ≤ error	0

ILO Mapping

ILO 8.6C

Evidencing

In your learning log section C, under ILO 8.6C, provide:

- Link to `task11_rl_controller/README.md`
- Link to `ot2_gym_wrapper.py`
- Link to `test_wrapper.py`
- Link to the RL training code
- Link to the RL testing code
- Link to the best individual model weights
- Link to the best group model weights
- GIF(s) of robot moving to target coordinates controlled by the RL agent
- Link to presentation slides with controller performance visualizations and documenting hyperparameter search results

Task 12: System Integration

In this task you will integrate your controllers (both PID and RL) with the computer vision pipeline to create a complete autonomous system. The goal is to enable the robot to automatically detect root tips in specimen plates and accurately inoculate them. You will need to transform pixel coordinates from the CV pipeline into robot coordinates.

Deliverables

- `task12_13_integration_benchmarking/README.md` with documentation of integration approach and coordinate transformation.
- Well documented code integrating the PID controller with the CV pipeline
- Well documented code integrating the RL controller with the CV pipeline
- GIF(s) showing both controllers autonomously inoculating root tips in multiple specimen plates

Client Requirements

The integrated system should meet the same accuracy requirements as the controller alone.

ILO Mapping

ILO 8.6D - 4 points

Evidencing

In your learning log section C, under ILO 8.6D, provide:

- Link to `task12_13_integration_benchmarking/README.md`
- Link to the integrated controller code PID
- Link to the integrated controller code RL
- GIF(s) of autonomous root tip inoculation with both controllers

Task 13: Performance Benchmarking and System Evaluation

In this task you will conduct a comprehensive performance evaluation of your integrated system. You will quantify the drop location accuracy relative to root tip locations and compare the performance of the PID and RL controllers (and between different RL or PID variants you developed). Your analysis should consider both accuracy and execution time, identify system limitations, and provide actionable recommendations for improvement to meet desired specifications.

Deliverables

- `task12_13_integration_benchmarking/README.md` with comprehensive documentation of the benchmarking approach and results, including:
  - Comparison of PID vs RL controller performance
  - Evaluation of different RL controllers or PID configurations (if applicable)
  - Accuracy metrics (e.g., drop location vs root tip location error)
  - Execution time and speed analysis
  - Visualizations of controller performance across defined metrics
  - Actionable recommendations for system improvements based on evaluation results
- Well-documented code implementing the performance benchmarking process
- Presentation summarizing benchmarking results and visualizations from the README

Client Requirements

Benchmark metrics that cover the following at a minimum:

- Accuracy and error metrics
- Execution time
- Actionable recommendations for improvement

ILO Mapping

ILO 8.6D - 5 points

Evidencing

In your learning log section C, under ILO 8.6D, provide:

- Link to `task12_13_integration_benchmarking/README.md`
- Link to the integrated controller code PID
- Link to the integrated controller code RL
- GIF(s) of autonomous root tip inoculation with both controllers
- Link to the benchmarking code
- Link to the presentation with benchmarking results

Task 14: Presentation

Present your creative brief solution that combines computer vision, robotics, and reinforcement learning (W8 Fri).

Deliverables

- A PDF file of your presentation slides.
- A recorded video of your presentation.

Client Requirements

- You have 15 minutes for your presentation. If you exceed 16 minutes, your mentor will stop you.

Contents:

- Problem definition (ILO 5.1)
- Overview of the proposed solution (ILO 8)
- How your solution will help the client (ILO 5.1)
  - Explain from a plant science point of view, not an AI point of view.
  - e.g “My solution can help plant scientists inoculate hundreds of plants automatically and find which genotypes are more resistant to diseases” etc.
- Results & Evaluation (ILO 8)
  - Your best pipeline combining computer vision and robotics
  - Show learning curves, metrics, images, masks, plots, benchmarking results
- Error analysis & Iteration (ILO 2.6)
  - During the block, you described and discussed your iterations in your worklog and learning log.
  - Select at least 3 iterations that you think were crucial for you to reach your best solution.
  - Discuss error analysis, your findings, your attempts to solve these errors, and the results of your attempts.
  - For instance: “I noticed discontinuities in my predicted root masks, which led to underestimations in root length predictions. To address this issue, I implemented X, Y, and Z, resulting in an improvement in the sMAPE score from 20% to 17%.” This explanation should be accompanied by relevant images, plots, and metrics to clearly illustrate the performance before and after the intervention.
- Assumptions
  - You need to explain all the assumptions you made during the design of your pipeline.
  - For example “I developed this pipeline assuming there will always be 5 plants in the Petri Dish.”.
- Limitations
  - You need to list all the limitations you noticed.
  - For example “I noticed my segmentation model’s performance decreases as the condensation increases. Therefore, my solution is suboptimal under heavy condensation.”
- Next steps (ILO 8)
  - Discuss what can be improved
- Summary

ILO Mapping

ILO 1.6, ILO 2.6, ILO 5.1, ILO 8

Evidencing

In your learning log section C,

ILO 1.6: For each criterion from A to E justify why you think you satisfy the criterion. For example under ILO 1.6C you can write “I was able to answer most of the questions during the presentation. Alican asked me the effect of patch size on model performance and I explained that higher patch size provides more context at the expense of computational cost”. Link to the recording & slides.

ILO 2.6: In your worklog, ensure that you log items where the “Most suited ILO” is ILO 2.6, with a minimum of two entries per week. In Section B (Reflection) of your learning log, address the question linked to ILO 2.6 each week. During your presentation, highlight at least three significant iterations related to ILO 2.6. For example: “I logged at least five work items each week, with a total of 50 work items for the entire project, as documented in my worklog. Each week, I summarized these items in Section B of my learning log. In my presentation (slides 8, 9, and 10), I showcased three key iterations that were most impactful.” Link to the recording & slides.

ILO 5.1: Summarize what you presented with regards to domain knowledge “In my presentation I explained how plant scientists struggle with X, Y, Z and how my solution provides A, B, C to …” and point to the slide numbers that are related. Link to the recording & slides.

ILO 8: This ILO will be evidenced by the previous ILOs. However, use your slides as supporting evidence. Following the example from Task 5 (ILO 8.5A):

“I trained a U-Net model for root segmentation, achieving an F1 score of 0.8021 on the validation set. I also developed inference code capable of processing raw images from Hades and generating the corresponding predicted masks. I presented the performance of this model in slides 6 and 7.”

,Applied Data Science and Artificial Intelligence @ Breda University of Applied Sciences
Contact : Frank Peters