PHENOSPEX
Smart Plant Analysis

Machine Learning Engineer – Take Home Assignment

Plant Counting and Localization Under Edge Constraints

Cagkan Gursoy

23.12.2025

Contents

1. Problem Definition

The goal of this work is to design a pipeline that **counts individual plants** and **estimates their centroid locations** from 2D and 3D sensor data. For a given input, the system outputs the total number of plants and the coordinates of each plant center.

The task is formulated as a **dense localization problem** rather than a bounding-box detection problem, focusing on accurate identification of plant centers. This formulation naturally supports heatmap-based regression approaches, where the model learns a continuous spatial representation of plant likelihood. Accordingly, two approaches are implemented: **a machine learning based solution** as the primary method and a **classical computer vision baseline** for comparison.

In addition to accuracy, the solution must satisfy practical deployment constraints, including CPU inference, offline operation, low memory usage and fast runtime, and a reproducible and lightweight training pipeline.

2. Dataset Description

The plant phenotyping dataset contains 2D top-down images with varying plant counts and grid configurations, including regular and irregular grid layouts. The dataset also includes background images with no plants and images containing empty pots only. The **2D modality** is used exclusively to prioritize computational efficiency, low inference latency, and CPU deployment.

An annotated dataset is constructed and split into **50** training images, **12** validation images, and **12** test images, while preserving the distribution of grid types across all splits. In addition, an augmented test set of **36** images is created for robustness evaluation.

2.1. Annotation Procedure and Assumptions

Plant centroid annotations are generated using a semi-automated annotation pipeline using OpenCV. Centroids are defined as 2D points (x, y), corresponding to the approximate stem base and assumed to align with the **center of the pot**. While more accurate centroid ground truth could be obtained from 3D point cloud data, this approximation is considered sufficient for the proposed approach.

3. ML Method: Architecture, Training Process, and Rationale

The ML approach is formulated as a **heatmap regression** problem and implemented using a lightweight **U-Net** architecture. This formulation is well suited to plant layouts with high visual similarity, where instances are defined primarily by their center locations rather than precise object boundaries. Given an input image, the model predicts a **single-channel heatmap** in which each plant center is represented by a **2D Gaussian peak**. Plant centroids are obtained by detecting **local maxima**, and the total plant count corresponds to the number of detected peaks.

Compared to bounding-box detectors (e.g., YOLO-style models), heatmap regression avoids **anchor design**, **bounding-box regression**, and **non-maximum suppression**, which are unnecessary for point-level localization. **Semantic segmentation** methods, while successful, need higher computational cost and **annotation complexity** without directly optimizing centroid accuracy. By directly supervising spatial plant locations, the proposed method provides a simpler training objective, **lower inference overhead**, and greater stability, making it suitable for **CPU deployment under edge constraints**.

3.1. Architecture Differences from Original U-Net

The proposed model follows the core **U-Net** structure but uses a simplified design. It employs fewer **encoder-decoder** stages and **reduced channel widths**, resulting in a **lower parameter count** and **faster CPU inference**. This compact design provides a favorable trade-off between **localization accuracy** and **inference efficiency** under strict latency and memory constraints. Unlike the original U-Net, which is typically trained for **semantic segmentation** using **pixel-wise classification losses**, the proposed model predicts a **single continuous heatmap** and is trained using **mean squared error loss** on **Gaussian targets**.

3.2. Training Procedure

In the training stage, ground-truth plant centroids are converted into **isotropic Gaussian kernels**, which are summed to form a single **ground-truth heatmap** per image. All inputs are resized to 512×512 in the model prior to training.

The network is trained using **mean squared error (MSE)** loss and optimized with the **Adam** optimizer. A **ReduceLROnPlateau** scheduler is used to stabilize convergence by lowering the learning rate when validation loss plateaus. **Early stopping** is applied to prevent overfitting, and best model checkpoint selection is based on validation loss.

Training is first performed without data augmentation to establish a baseline. Augmented training then incorporates **rotations**, **occlusions**, and **noise** to improve robustness and generalization, with evaluation on an augmented test set. Finally, due to the limited number of irregular grid samples, an **oversampling strategy** is used to mitigate dataset imbalance.

4. Classical Computer Vision Baseline

A **classical computer-vision pipeline** is implemented as a baseline, relying on hand-crafted image processing steps to **remove background structure** and **segment plant leaf** regions to compute centroids.

The pipeline starts with **region cropping** to remove irrelevant background areas outside the plant grid. Next, white shadows and table surfaces are suppressed using **color space thresholding**. Plant regions are then segmented via **grayscale conversion**, **denoising**, and **Otsu thresholding**. In the post-processing stage, small noise components are removed and fragmented plant regions are merged using **morphological operations**. Finally, **connected components** are extracted, and plant centroids are computed from the resulting components.

5. Results

Performance is evaluated using the following metrics:

- **Localization error:** The **mean Euclidean distance (in pixels)** between matched predicted and ground-truth centroids. This metric is computed only over matched pairs within a **5 px threshold** for the ML approach, resulting in values bounded by this threshold.
- **F1 score:** Calculated between predicted centroids that are **greedily matched** to ground-truth centroids within a fixed distance threshold.
- **Count error:** Defined as the **mean absolute error (MAE)** between predicted and ground-truth plant counts, computed per image.

5.1. Effect of Hyperparameters on the ML Model

A set of training runs was conducted to analyze the effect of key hyperparameters, including the **Gaussian heatmap sigma (σ)**, the **oversampling weight (OSW)** for irregular grid samples, and the **sampling multiplier (SM)**.

The **Gaussian sigma** has the strongest influence on performance, with larger values outperforming smaller ones. In addition, applying **OSW** during training helps mitigate dataset imbalance and improves counting marginally.

Based on the test set results, the configuration **($\sigma = 3.0$, OSW = 25, SM = 2)** is selected. Although this setting shows a minor reduction in localization accuracy compared to the **OSW = 12** configuration, it achieves a lower counting error (MAE) and a modestly higher F1 score. This localization trade-off is acceptable given the improved true-positive detection rate.

Table 1: **Test-set results of key ML experiments.**

σ	OSW	SM	F1	Loc. Err (0–5 px)	Count Err (MAE)
3.0	1	1	**0.973**	1.62	1.25
3.0	25	2	0.971	1.61	**0.83**
3.0	12	2	0.968	**1.53**	2.08
1.0	12	2	0.750	2.38	14.33

5.2. Robustness and Augmented Test Evaluation

Robustness is evaluated by testing the selected model on both the **test set** and an **augmented test set**, comparing training performed **with** and **without data augmentation**. Models trained without augmentation generalize poorly to the augmented test set, while models trained with augmentation maintain strong performance across both sets. This highlights the importance of augmentation for robust generalization.

The applied augmentation is limited to horizontal and vertical flips, reflecting expected real-world acquisition conditions. However, the performance degrades for unseen rotations (e.g., 45°, as observed in the demo).

5.3. Comparison with Classical Baseline

The ML model is compared against the classical computer-vision baseline on both the test and augmented test sets. The classical method performs poorly under 5 px matching threshold, which is expected due to the

Table 2: **Test and augmented test performance with and without data augmentation.**

	Test			Test-Aug		
Aug.	**F1**	**Loc. Err (0–5 px)**	**Count Err (MAE)**	**F1**	**Loc. Err (0–5 px)**	**Count Err (MAE)**
No	0.971	1.71	0.83	0.404	4.07	25.81
Yes	**0.984**	**1.58**	**0.58**	**0.957**	**1.88**	**1.64**

mismatch between pot-based ground truth and leaf-based segmentation. Even when the matching threshold is relaxed to 50 px, the classical approach remains **substantially inferior** to the ML model.

Table 3: **Best ML model vs. classical baseline under different matching thresholds.**

	Test		Test-Aug	
Method	**F1 Score**	**Loc. Err (px)**	**F1 Score**	**Loc. Err (px)**
U-Net–Tiny (best)	**0.983**	**1.57**	**0.959**	**1.81**
Classical CV (5 px)	0.043	3.08	0.013	3.24
Classical CV (50 px)	0.624	21.31	0.188	23.49

5.4. Runtime Comparison

During training, **GPU acceleration** is used to reduce runtime. On **CPU**, a single epoch takes approximately **3:30 min**, while on **GPU** it takes about **1:30 min**. Peak GPU training usage is **4.7 GB of VRAM** and approximately **1.2 GB of system RAM**.

During inference on **CPU**, the **classical pipeline** processes an image in **0.15 s**, whereas the **ML model** requires **0.60 s** using **PyTorch**. After **ONNX** export, inference time is reduced to **0.25 s**, remaining only slightly slower than the classical approach while providing substantially improved **accuracy** and **robustness**.

6. Deployment Considerations

For deployment, the trained model is exported to **ONNX** and validated. The exported model has a size of **7.35 MB**, making it suitable for resource-limited environments. The average runtime of the ONNX model is **0.23 s per image**, which is sufficient for near real-time operation.

Memory usage is measured using **resident set size (RSS)**. Loading the ONNX model increases memory usage from approximately **500 MB** to **540 MB**. During inference, peak memory usage reaches about **900 MB**, primarily due to the runtime environment, image data, and intermediate tensors, rather than the model itself.

Overall, the small model size, reasonable CPU runtime, and framework-independent ONNX format make the proposed method suitable for practical deployment. Further memory and runtime reductions may be achieved using a lighter runtime, such as a **C++-based ONNX Runtime**.

7. Limitations and Future Improvements

The proposed approach relies on some simplifying assumptions, notably that plant centroids are approximated by the pot center. This ignores both depth information and the true geometric centroid of the plant. The model is trained on a limited dataset with mostly regular grid layouts and visually similar plants. Although data augmentation improves robustness, this can still restrict generalization to other plant species, growth stages, and grid configurations.

Robustness to image rotations is limited by the current augmentation policy. In particular, for rotations other than horizontal and vertical flips, the model shows reduced performance. Another limitation is the dependence on pot visibility. Because annotations are defined at the pot center, the model may primarily learn pot-specific visual cues. As a result, performance can decline when pots are occluded, even if the plant remains visible, leading to false negatives. Conversely, false positives may occur when a pot is present without a plant. These effects are observed in samples containing pots without plants, where rare false predictions are produced.

Future work can include using 3D data to obtain accurate centroid annotations and expanding to larger and more diverse datasets to improve generalization under accuracy, runtime, and resource constraints. In addition, extending the augmentation policy with additional rotations can further improve robustness to variations in image orientation. Finally, to reduce pot-only detections, introducing negative annotation or penalties for empty pots, applying post-processing filters, or adopting a plant segmentation model are promising directions.