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 objective of this work is to design a pipeline that **counts individual plants and estimates their centroid locations** from **2D and 3D sensor data**. Given an input, the system must output the **total number of plants**, and the **coordinates** of each plant's center.

This task is formulated as a **dense localization problem** rather than a **bounding-box detection problem**. Instead of predicting object extents, the focus is on accurately identifying **plant centers**. This formulation naturally supports **heatmap-based regression approaches**, where the model learns a **continuous spatial representation of plant likelihood**.

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

For the assignment, two approaches are implemented: **a machine learning-based solution** as the primary method and a **classical computer-vision baseline** for comparison.

2. Dataset Description

This work utilizes the provided phenotyping dataset with varying **grid configurations and plant counts**, focusing exclusively on **2D top-down images**. The **2D modality** was selected to prioritize **computational efficiency, low inference latency, and CPU-only deployment**.

An annotated dataset was built and split into **50 training images**, **12 validation images**, and **12 test images**, preserving the **distribution of grid setups** across all splits. An additional **augmented test set of 36 images** is constructed 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** in which instances have **high visual similarity** and are primarily defined by their **center locations** rather than precise object boundaries. Given an input image, the model predicts a **single-channel heatmap** where each plant center is represented by a **2D Gaussian peak**. Plant centroids are obtained by detecting **local maxima** in the predicted heatmap, 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** approaches, while expressive, introduce **higher computational cost** and **annotation complexity** without directly optimizing centroid accuracy. By directly supervising **spatial plant locations**, the proposed method yields a **simpler training objective**, **lower inference overhead**, and **greater stability**, making it well suited for **CPU-only deployment under edge constraints**.

3.1. Architecture Differences from Original U-Net

The proposed model includes the core **U-Net encoder-decoder structure with skip connections** but is **simplified**. This variant uses **fewer encoder-decoder stages** and **reduced channel widths**, resulting in a **significantly lower parameter count** and **improved CPU inference speed**. This **compact design** achieves 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 outputs a **single continuous heatmap** and is trained using **mean squared error loss** on **Gaussian targets**.

3.2. Training Procedure

Ground-truth plant centroids are converted into **isotropic Gaussian kernels** centered at each centroid, which are summed to form a single **ground-truth heatmap** per image. All images are resized to **512**$\times$**512** in the model prior to training.

The network is trained to regress heatmaps using **mean squared error (MSE)** loss and optimized with the **Adam** optimizer. To **stabilize convergence**, a **ReduceLROnPlateau** learning rate scheduler is employed, reducing the learning rate when validation loss plateaus. **Early stopping** is applied to mitigate overfitting and avoid unnecessarily long training runs. **Best model checkpoint selection** is based on validation loss.

Initial training is performed **without data augmentation** to establish a baseline. Subsequently, augmented training is done using **rotations**, **occlusions**, and **noise** to improve **robustness and generalization**, and evaluated on an augmented test set. Due to the **limited number of irregular grid samples**, an **oversampling strategy** is applied during training to increase their contribution and mitigate dataset imbalance.

4. Classical Computer Vision Baseline

A **classical computer-vision pipeline** is implemented to provide an alternative **baseline**. This baseline relies on **manually tuned thresholds** and **morphological parameters**. The pipeline begins with **region cropping** to remove **irrelevant background areas** outside the **plant grid**. **Bright glare** and **table surfaces** are then suppressed using **color-space thresholding**, followed by **binary segmentation of plant regions** via **grayscale conversion**, **denoising**, and **Otsu thresholding**. In a **post-processing stage**, **small noise components** are removed and **fragmented plant regions** are merged using **morphological operations**. Finally, **connected components** are identified, and **plant centroids** are computed from the resulting **components**.

5. Results

Performance is evaluated using:

- **F1 score:** Where **predicted centroids** are **greedily matched** to **ground-truth centroids** within a **fixed distance threshold**.
- **Localization error: Mean Euclidean distance (in pixels)** between **matched predicted and ground-truth centroids**. The **localization error** is reported only over **matched pairs** within a **5 px threshold for ML approach**, resulting in values **bounded by this threshold**.
- **Count error** is 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 assess 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**.Also, applying OSW during training helps mitigate dataset imbalance and improves overall results.

Further experimentation could yield better results; however, based on the current test set results, the configuration **($\sigma = 3.0$, OSW = 25, SM = 2)** is selected. While this setting shows a slight reduction in localization accuracy compared to the **OSW = 12** configuration, it achieves a **lower counting error (MAE) and higher F1**, where the minor 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 final model on both the **test set** and the **augmented test set**, comparing training conducted **with** and **without data augmentation**. Models trained **without data augmentation** generalize poorly to the augmented test set. In contrast, models trained **with data augmentation** maintain strong performance across both test conditions. These results demonstrate that data augmentation is critical for achieving robustness and reliable generalization.

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**

5.3. Comparison with Classical Baseline

Comparison of the **best ML model** with a **classical CV baseline** on both the **test** and **augmented test set** was done. The **classical method** performs **poorly** under a **5 px matching threshold**, which is expected given the **mismatch between pot-based ground truth and leaf-based segmentation**. However, relaxing the threshold to **50 px** still results in **substantially inferior performance** compared 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

For training, **GPU acceleration** was used to reduce training time. On CPU, a single epoch required approximately **3:30 min**, whereas in GPU it is **1:30 min**. Peak resource usage during GPU training was **4.7 GB of VRAM** and approximately **1.2 GB of system RAM**.

For inference, on **CPU**, the **classical pipeline** runs in **0.15 s per image**, while the **ML model** requires **0.60 s** with **PyTorch** and **0.25 s** after **ONNX** export, remaining only slightly slower while offering substantially improved accuracy and robustness.

6. Deployment Considerations

For **deployment**, the trained model is exported to **ONNX** and validated. The exported model size is **7.35 MB**, making it suitable for **resource limited environments**. **Runtime evaluation** using **ONNX Runtime on CPU** shows an **average inference time** of **0.23 s per image**, which is adequate for **near real-time runs**. **Memory usage** was measured using **resident set size (RSS)**. Loading the ONNX model increased memory usage from about **500 MB** to **540 MB**. During inference, **peak memory usage** reached approximately **900 MB**, mainly 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** could be possible using a **lighter runtime** (e.g., C++ ONNX Runtime).

7. Limitations and Future Improvements

The **proposed approach** relies on **simplifying assumptions**, notably that **plant centroids** are approximated by the **pot center**, which 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**, which can restrict **generalization** to other **plant species**, **growth stages**, and **grid configurations**, even though **data augmentation** improves **robustness**.

Another limitation arises from the **dependence on pot visibility**. Because annotations are defined at the **pot center**, the model may primarily learn **pot-specific visual cues**. Consequently, 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 rarely false predictions are produced.

Future work can include using **3D data** to obtain **accurate centroid annotations**. Expanding to **larger and more diverse datasets** may further improve **generalization** under **accuracy**, **runtime**, and **resource** constraints. Additionally, introducing **negative annotations** for empty pots, applying **post-processing filters**, or adopting a **plant segmentation model** could mitigate pot-only detections.