UNIVERSITÀ DEGLI STUDI DI UDINE
HIC SUNT FUTURA

DEPARTMENT OF MATHEMATICS, COMPUTER SCIENCE AND PHYSICS

ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN | GRAZ

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND CYBERSECURITY

MASTER THESIS IN
ARTIFICIAL INTELLIGENCE & CYBERSECURITY

Artificial Intelligence for Precision Agriculture: Semantic Segmentation of Crop Fields Using Deep Learning

CANDIDATE
Federico Dittaro

SUPERVISOR
Prof. Giuseppe Serra

CO-SUPERVISORS
Dott. Alex Falcon
Dott.ssa Beatrice Portelli
Prof. Kyamakya Kyandoghere

Academic Year 2024-2025

Institute Contacts
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
https://www.dmif.uniud.it/

To my family,
for their constant support, and to myself,
for never giving up.

Abstract

Accurate estimation of vegetation cover is essential for agricultural monitoring, particularly in complex field conditions such as intercropping systems where multiple species coexist and overlap. Traditional manual methods for assessing vegetation cover are time-consuming, subjective, and difficult to scale. In this work, we present an automated approach based on deep learning-driven semantic segmentation to estimate class-specific vegetation cover directly from top-down RGB images tha can be deployed even on resource-constrained devices such as smartphones. We frame the problem as a multi-class segmentation task, distinguishing between soil, crop and weed species. Our pipeline integrates different training strategies and it is based on the Convolutional Neural Networks (CNNs). We systematically compare six Encoder-Decoder combinations, including MobileNetV2 and EfficientNet-B0 as Endocders and UNet, UNet++ and DeepLabV3 as Decoders. Models are evaluated on the datasets used for training and on a post-evaluation dataset of annotated field images collected from a controlled cropping experiment in the University of Udine, using both quantitative metrics (Precision, Recall, IoU and F1-score) and qualitative visual analysis. Results show that all models achieve strong performance, with EfficientNet-B0 paired with the UNet achieving the highest metric scores. However, also other models trained with different loss functions deliver competitive results especially on the underrepresented classes. Ablation studies confirm the importance of exploring different training strategies since it allows to detect the most balanced and robust segmentation performance. This work demonstrates that deep learning-based semantic segmentation can effectively automate vegetation cover estimation in agricultural settings. The proposed system paves the way for deploying cover estimation tools in the field, with potential applications in crop monitoring, decision support, and sustainable farm management.

Contents

1

Introduction

1.1 Motivating Scenario

Agriculture today is under pressure to produce more food with fewer resources while minimizing its ecological impact. Weeds are among the major constraints to crop productivity, as they compete directly with crops for water, light, and nutrients [39]. So consequently weed control is indispensable in modern farming. Traditional approaches rely heavily on blanket herbicide application. Although effective at suppressing weeds, this method is indiscriminate: herbicides are applied to the entire field regardless of whether crops, weeds, or bare soil occupy a given location. Such practices result in unnecessary chemical use, increased production costs, and negative environmental consequences including soil contamination, water pollution, and the emergence of herbicide-resistant weed species. Manual weeding, while precise, is too labor-intensive to be viable at scale [14].

Currently, agronomists and farmers often estimate the composition of plants and weeds in a field through manual "sampling quadrats," in which a square frame is placed on the soil and the vegetation within it is visually assessed. While well established, this method demands expertise, is time-consuming, and is subject to interpersonal variability——different observers may produce different estimates under the same conditions. Modern AI-based image analysis offers a path to automate this process, reduce human bias, and enable in-field assessments using mobile devices or drone imagery.

Moreover, in real fields the problem is not simply the presence of weeds, but their close spatial and visual similarity to crops, especially at early growth stages. For example, in a field of sugar beets or maize, weeds often emerge simultaneously and share similar shapes, textures, and colors with the crop seedlings. Distinguishing one from the other is a non-trivial challenge for both human laborers and automated systems. A system that detects only weeds without explicitly recognizing where crops are risks false positives that could lead to unintentional crop damage. Conversely, a system that detects only crops may fail to identify invasive weeds that grow in between. This challenge highlights the importance of crop–weed segmentation, where each pixel in a field image is classified as belonging to a specific type of crop, weed, or background soil [23]. Unlike simple weed detection, segmentation provides a holistic understanding of the field allowing the precision farming systems to act selectively enabling for example an autonomous robot to navigate intelligently between crop plants.

The benefits extend beyond operational accuracy. By leveraging crop–weed distinction, farmers can re-

duce input costs by minimizing wasted herbicide, safeguard yield by avoiding crop loss from mechanical damage, and reduce environmental pollution from over-application. From a broader perspective, improved weed management through segmentation supports the transition toward sustainable agriculture, balancing productivity with environmental stewardship.

1.2 Objectives

The primary objective of this work is to automate the estimation of vegetation cover from images captured in real agricultural fields. Specifically, we focus on achieving a balance between accuracy and efficiency in the semantic segmentation of weeds, crop species, and soil.
Since we are analyzing complex images, we investigate deep learning (DL) models since they are particularly effective for understanding the content of such images, whereas older approaches often struggle to capture this complexity. Specifically, we focus on achieving a balance between accuracy and efficiency in the semantic segmentation of weeds, crop species, and soil. To this end, we systematically evaluate a diverse set of convolutional neural networks (CNNs), experimenting with different Encoder–Decoder architectures and training strategies to identify the most effective approaches. We decided to use CNNs because they are generally easier to fine-tune when only limited samples are available, thanks to their inductive biases. This tendency, together with training strategies that help mitigate class imbalance and improve efficiency, provides a practical justification for concentrating on CNNs. A major challenge lies in the datasets themselves. Publicly available datasets are not only limited in size and balance but also lack the variety and realism of actual intercropping systems. For example, crops such as soybeans and lentils——common in Italy——are scarcely represented in many public datasets, which are often dominated by images of isolated plants with little overlap. These conditions differ significantly from the complexity of real agricultural fields and create additional hurdles for accurate model training. By addressing these issues of dataset representativeness, class imbalance, and computational efficiency, this work demonstrates that deep learning offers a scalable, accurate, and reproducible solution for vegetation cover assessment that can be deployed even on resource-constrained devices such as smartphones.

1.3 Summary of contributions

The main contributions of this work encompass experimental design, evaluation, and practical implementation, all within the context of semantic segmentation for vegetation cover estimation in agricultural settings. The central goal is not merely to implement deep learning methods, but to develop approaches that are efficient and mobile-friendly, paving the way for future applications such as a vegetation cover assessment app. By focusing on both accuracy and computational efficiency, this work ensures that the proposed methods are also practically deployable in real-world agricultural scenarios. The contributions can be summarized by the following points:

- **Automated pipeline and experimental reproducibility**: we develop a fully automated and reproducible experimental pipeline to handle the complete workflow of model training, evaluation, and analysis. The pipeline ensures consistency, minimizes human error, and supports extensive logging, checkpointing, and post-training evaluation. It integrates patch extraction strategies,

training monitoring through Weights & Biases (wandb), and systematic model comparison, enhancing both the reproducibility and scalability of experiments.

- **Comprehensive evaluation of semantic segmentation models**: we conduct a comprehensive evaluation comparing various state-of-the-art semantic segmentation models for the task of vegetation cover estimation in complex intercropping systems. We systematically test a diverse set of architectures (UNet, UNet++, and DeepLabV3) and backbones (MobileNetV2 and EfficientNet-B0) comparing them in terms of performance, parameter efficiency, and suitability for the specific segmentation task.

- **Loss function analysis for unbalanced data**: we conduct an extensive comparative study of multiple loss functions, analyzing their impact on model performance under severe class imbalance. Specifically, we evaluate Cross-Entropy Loss, Weighted Cross-Entropy (WCE), Dice Loss, and Focal Loss individually.

- **Qualitative and quantitative evaluation of segmentation performance**: we rigorously assess model performance using both quantitative segmentation metrics—including per-class and aggregated IoU, F1, precision, and recall scores—and qualitative visual analysis.

2

Background

This chapter provides the theoretical foundations necessary to understand the methods and experiments presented in the following chapters. It begins with an overview of computer vision, introducing the key concepts and processing stages involved in enabling machines to interpret visual information. The discussion then focuses on semantic segmentation, highlighting its role in pixel-level image analysis and detailing the specific challenges that arise when applied to agricultural contexts. The chapter proceeds with an introduction to Convolutional Neural Networks, the core deep learning models used in this work. Their fundamental principles and architectural components are described, with particular emphasis on their application to semantic segmentation tasks in agriculture. Subsequently, we present the concepts of encoders and decoders, which together form the backbone of modern segmentation architectures. Finally, we provide a more detailed description of the specific encoder backbones (MobileNetV2, EfficientNet-B0) and decoder architectures (UNet, UNet++, DeepLabV3) that are experimentally evaluated in this thesis.

2.1 Computer vision

Computer vision is a subfield of artificial intelligence (AI) and computer science that focuses on enabling machines to interpret and understand visual information from the physical world. It involves the development of algorithms and systems that can acquire, process, analyze, and extract meaningful information from digital images or video streams. Unlike human vision, which leverages biological perception and cognitive processes, computer vision relies on computational models to replicate similar capabilities, such as object recognition, scene understanding, motion tracking, and semantic segmentation. The primary goal of computer vision is to automate tasks that the human visual system can perform. These tasks range from low-level image processing, such as edge detection and filtering, to high-level reasoning tasks, such as recognizing objects in complex environments or interpreting human emotions from facial expressions.
Computer vision systems typically involve several stages of processing:

1. **Image Acquisition**: capturing visual data using sensors such as cameras, LiDAR, or depth sensors.

2. **Preprocessing**: enhancing image quality through techniques like noise reduction, normalization, and contrast adjustment.

3. **Feature Extraction**: identifying and representing significant patterns in images using local descriptors or learned features via deep networks.

4. **Object Recognition and Detection**: classifying and locating objects within an image, commonly achieved through deep learning-based models.

5. **Segmentation**: dividing an image into meaningful regions, often used for scene understanding or medical imaging.

6. **Post-Processing and Decision Making**: applying higher-level reasoning, such as tracking or action planning, based on the extracted information.

Computer vision is widely applied across numerous domains. In the automotive sector, it serves as a technology for autonomous vehicles, enabling functionalities such as lane detection, obstacle avoidance, and pedestrian recognition to ensure safe navigation [16] [31] [28]. In healthcare, computer vision techniques play a role in medical imaging by assisting in the detection of tumors, analyzing radiographs, and supporting clinical diagnostics [2] [52]. Security and surveillance systems leverage computer vision for facial recognition, anomaly detection, and real-time activity monitoring, enhancing both public safety and access control [61] [12]. In industrial automation, vision systems contribute to quality control processes and guide robotic manipulators during assembly or manufacturing operations [41] [4]. Agriculture also benefits from computer vision through applications such as crop monitoring, disease detection, and yield estimation, which improve resource management and optimize farming practices [67] [9] [76]. Additionally, the field is integral to augmented and virtual reality applications, where real-time scene understanding enables immersive user experiences.
Figure 2.1 illustrates the distinctions among several common computer vision tasks. Image classification (top left) determines which classes are present in an entire image. Object detection (top right) goes a step further by identifying and localizing individual objects using bounding boxes. In contrast, semantic segmentation (bottom left) assigns a class label to every pixel, delivering a dense, pixel-level understanding of the scene. Finally, instance segmentation (bottom right) builds on semantic segmentation by distinguishing between separate instances of the same class.

2.2 Semantic segmentation

Semantic segmentation is a computer vision task that provides a dense, pixel-level understanding of an image. Unlike image classification, which assigns a single label to an entire image, or object detection, which predicts bounding boxes for objects, semantic segmentation assigns a class label to every pixel in the image. Formally, semantic segmentation can be defined as a mapping function:

$$f : \mathbb{R}^{H \times W \times C} \longrightarrow \{1, 2, \ldots, K\}^{H \times W}$$

where H and W represent the image dimensions, C the number of channels (e.g., 3 for RGB), and K the number of semantic classes. The output is a label map where each pixel is assigned to one of the K

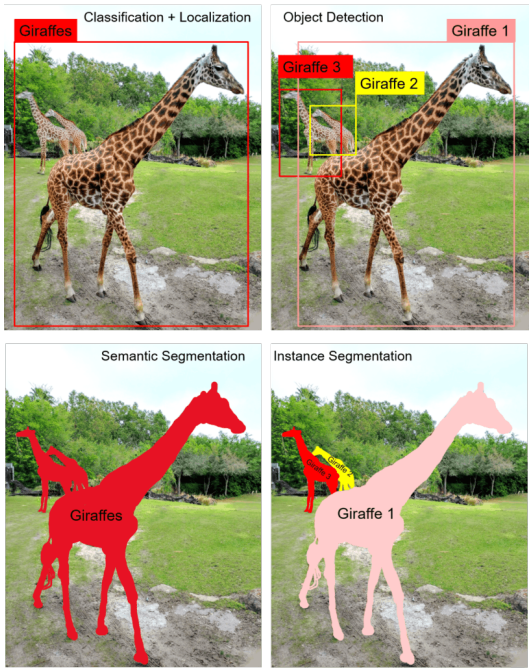


Figure 2.1: Comparison of common cumputer vision tasks.

categories. Semantic segmentation can be further categorized into three main types:

- **Binary Segmentation**: the goal is to separate pixels into two categories: foreground and background. This is common in tasks such as medical imaging or object extraction from backgrounds. For example, in plant-related imagery, binary segmentation could be used to distinguish between vegetation and non-vegetation (i.e., soil vs. plant).

- **Multiclass Segmentation**: each pixel belongs to exactly one class out of multiple possible categories. For example, in urban scene segmentation, pixels may represent road, sky, car, pedestrian, or building, but each pixel is assigned only one class. In an agricultural setting, this might involve classifying pixels as crop species, weed species, or soil.

- **Multilabel Segmentation**: a pixel can belong to multiple classes simultaneously. This is less common but useful in scenarios with overlapping features, such as detecting transparent weeds over crops, or labeling both a plant type and a semantic attribute like disease presence.

In this work, we focus on multiclass semantic segmentation, aiming to differentiate between plants, weeds, and soil in field images. This involves assigning each pixel to one of the specific categories: weeds, soil, or the corresponding class for each type of plant present.

2.2.1 Semantic segmentation in agriculture

Semantic segmentation has become a useful tool in precision agriculture due to its ability to provide detailed, pixel-level classification of agricultural imagery. Unlike traditional image analysis methods that rely on handcrafted features [24] [53], semantic segmentation powered by deep learning offers an automated and scalable solution for understanding complex agricultural environments. Agricultural fields pose several challenges for semantic segmentation, stemming from the complex and dynamic

nature of these environments. One major issue is *high intra-class variability*, as crops and weeds can differ significantly in appearance based on growth stage, health conditions, and environmental factors. On the other hand, there is *low inter-class variability*, since crops and weeds often share similar colors, shapes, and textures, especially during early growth stages, making it difficult to distinguish between them. Additionally, *dynamic environmental factors* such as varying lighting conditions, soil moisture, and the presence of crop residues further complicate visual consistency across images [44]. Finally, the problem of *class imbalance* arises because soil pixels typically occupy a much larger portion of the image than crops or weeds, requiring specific strategies to handle this skewed distribution effectively [7].
Despite these challenges, semantic segmentation is typically applied for tasks like:

- Weed Management: pixel-wise segmentation enables selective herbicide application, reducing chemical usage and minimizing environmental impact [65].
- Yield Estimation: accurate crop segmentation allows for better plant counting and biomass estimation [1].
- Disease and Stress Detection: early detection of disease symptoms on leaves can be achieved through high-resolution segmentation [58].
- Automation in Robotics: autonomous weeding robots rely on accurate segmentation maps to plan safe paths and perform precision mechanical weeding [63].

Other works

Traditional vegetation cover estimation tools, such as Canopeo [55], rely on simple color index thresholds. While effective for measuring total green cover in a binary vegetation/non-vegetation sense, Canopeo is not suited for more detailed multi-class vegetation-cover estimation or differentiation among plant species. Semantic segmentation addresses this limitation by enabling pixel-level classification for each species. For example, [50] demonstrated that a CNN trained exclusively on sole-crop images of pea and oat could generalize to intercropped plots, achieving an IoU of about 66% per class and providing reliable species-specific cover estimates. Similarly, [32] applied semantic segmentation to quantify fractional cover across multiple plant species, underscoring its value in ecological monitoring.
However, segmentation-based cover estimation must contend with class imbalance, particularly when minority classes such as weeds are underrepresented. To address this, [19] incorporated Dice loss to improve segmentation of minority categories, while [57] enhanced performance by augmenting real datasets with synthetic weed imagery. Together, these techniques improve per-class accuracy, ensuring that cover metrics remain robust and representative across all species present.

2.3 Convolutional Neural Networks

Convolutional Neural Networks [38] are a class of deep neural networks designed to process data with a grid-like topology, such as images. They have become the foundation of modern computer vision tasks, including image classification, object detection, and semantic segmentation, due to their ability to learn hierarchical feature representations directly from raw pixel data. Unlike traditional fully connected

neural networks, CNNs leverage three concepts—local connectivity, weight sharing, and translation invariance—to efficiently process high-dimensional input images. The fundamental building block of a CNN is the convolutional layer, which applies a set of learnable filters (kernels) to extract local features such as edges, textures, and shapes. As the network depth increases, higher layers capture more abstract and complex patterns, enabling semantic understanding of entire scenes. A typical CNN architecture for image processing consists of several stages:

- **Convolutional Layers**: perform convolution operations to detect spatial features using learnable filters that respond to local patterns. Given an input tensor $\mathbf{X} \in \mathbb{R}^{H \times W \times C_{in}}$ where H, W and C_{in} represent the height, width, and number of input channels, the convolution filter $\mathbf{K}_i \in \mathbb{R}^{k \times k \times C_{in}}$ produces an output feature map $\mathbf{Y}_i \in \mathbb{R}^{H' \times W'}$ such that:

$$\mathbf{Y}_i = \mathbf{X} * \mathbf{K}_i + b_i$$

  where $*$ denotes the discrete convolution operator, and $b_i \in \mathbb{R}$ is a learnable bias term. The convolution operation can highlight specific patterns such as edges, textures, or more complex shapes.

- **Activation Functions**: after the convolution operation in a neural network, an activation function is applied to introduce nonlinearity into the model. Nonlinear activation functions are important because, without them, the network would behave like a linear model regardless of its depth, limiting its ability to capture complex patterns in the data. Some of the most commonly used activation functions include:

  - *Rectified Linear Unit (ReLU)*: defined as $f(x) = max(0, x)$ ReLU is widely used in modern neural networks due to its simplicity and computational efficiency. It also helps mitigate the vanishing gradient problem [51], allowing deep networks to train more effectively. Variants like Leaky ReLU and Parametric ReLU (PReLU) further improve performance by allowing a small, non-zero gradient when the input is negative.
  - *Sigmoid*: given by $\sigma(x) = \frac{1}{1+e^{-x}}$ the sigmoid function maps input values to a range between 0 and 1. It is useful for probabilistic interpretations, such as in the output layer of binary classification tasks. However, it can suffer from vanishing gradients for very large or very small inputs.
  - *Hyperbolic Tangent (tanh)*: defined as $tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ this function maps inputs to the range $[-1, 1]$ and is zero-centered, which often leads to faster convergence compared to the sigmoid function but it can suffer from vanishing gradients too.

- **Pooling Layers**: are used to downsample feature maps, which reduces the spatial dimensions (width and height) of the input while retaining the most important information [5]. This helps to reduce the number of parameters and computational cost, and also increases the network's robustness to small translations, rotations, and other spatial variations in the input. The most common types of pooling are: *Max pooling* which selects the maximum value from a defined window, *Averege pooling* which computes the average value of elements in the window and *Global*

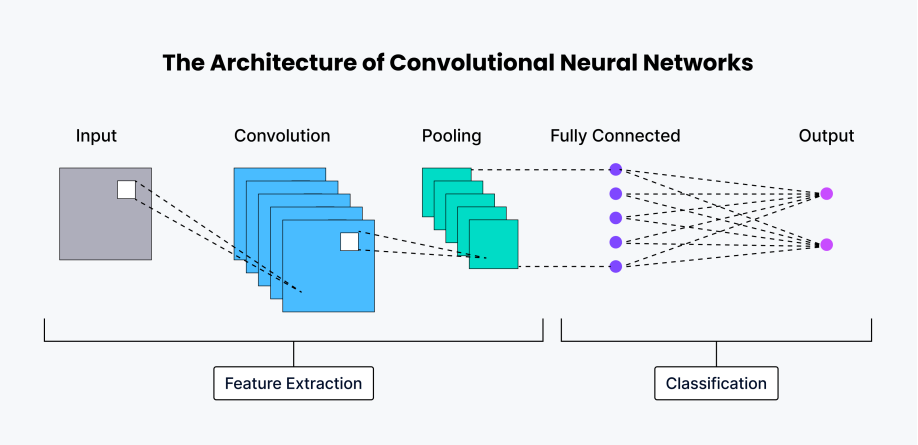


Figure 2.2: Basic Convolution Neural Network architecture.

pooling that reduces each feature map to a single value by taking the max or average over the entire spatial dimension.

- **Fully Connected Layers**: are used in neural networks to aggregate features learned by previous layers and perform tasks such as classification or regression. Each neuron in an FC layer is connected to all activations from the previous layer, allowing the network to combine features globally. Mathematically, the operation of a fully connected layer can be expressed as $\mathbf{y = f(Wx+b)}$ where x is the input vector, W is the weight matrix, b is the bias vector and f is the activation function.
- **Normalization and Regularization**: are techniques that improve training stability, convergence speed, and generalization in neural networks by applying techniques like batch normalization and dropout.

The strength of CNNs lies in their ability to automatically learn feature hierarchies, eliminating the need for handcrafted feature extraction. This property makes CNNs particularly suitable for agricultural applications, where visual variability is high and manual feature engineering is impractical.
Figure 2.2 illustrates a basic CNN architecture.

2.3.1 CNN-based semantic segmentation in agriculture

Convolutional Neural Networks have emerged as an approach for semantic segmentation in agricultural contexts, owing to their capacity to automatically learn hierarchical visual features from raw images. Unlike traditional computer vision techniques, which relied heavily on handcrafted descriptors such as color indices or texture filters, CNNs provide an end-to-end learning paradigm where relevant patterns are extracted directly from data. This property is particularly valuable in agriculture, where visual variability is high and manually designing robust features across different growth stages, lighting conditions, and soil backgrounds is impractical. In semantic segmentation, CNNs are typically deployed within encoder–decoder architectures, where the encoder progressively reduces the spatial resolution while extracting increasingly abstract features, and the decoder reconstructs a dense, pixel-wise prediction map. Well-known segmentation models such as U-Net, U-Net++, and DeepLabV3 have been widely adopted in crop–weed classification tasks, often combined with lightweight backbones (e.g., MobileNetV2, EfficientNet-B0) to ensure efficiency on edge devices used in precision farming. These ar-

chitectures enable the simultaneous classification of crops, weeds, and soil, addressing the challenge of their visual similarity at early growth stages. The next sections will go deeper on these topics.

Other works

Convolutional neural networks have become the standard approach for semantic segmentation in agricultural settings. For instance, lightweight CNNs have enabled real-time crop–weed segmentation on robotic platforms [49] achieving strong generalization across fields with limited annotated data. Such three-class models (crop, weed, background) demonstrate CNNs' potential for field robotics. Multiclass segmentation has also been explored, such as in barley and oilseed radish fields with VGG16-based networks (66% IoU) [18], and UAV-based U-Nets for alpine meadows, which achieved over 84% accuracy [32], underscoring CNNs' capacity for fine-grained vegetation classification using spatial texture cues.
Advancements in CNN architectures have further improved segmentation accuracy. Multi-level feature fusion modules enhance contextual understanding [29], while networks such as EPAnet leverage attention and dual decoders to segment bean seedlings and weeds, achieving notable IoU gains through combined cross-entropy and Dice loss functions [19]. Transfer learning is particularly effective in low-data regimes: U-Nets with ImageNet-pretrained encoders such as Inception ResNetV2 have reached F1 scores of 96.8% in multiclass weed detection [23]. Similarly, other studies confirm that pretrained CNNs significantly improve performance in data-limited agricultural segmentation [17]. Data augmentation also remains essential, with strong augmentation strategies boosting accuracy by over 14% [72].
Recent work has proposed hybrid CNN approaches for agricultural imagery. Mohammadi et al. (2022) introduced a deep semantic segmentation algorithm combining U-Net with K-means subtractive clustering, enabling precise crop–weed separation even under high weed density. Their method achieved up to 99.19% accuracy, outperforming traditional K-means and superpixel approaches, with a true-positive rate of 0.995 for crops and a true-negative rate of 0.898 for weeds. This integration of semantic segmentation with clustering offers robust performance in complex field conditions [67]. In parallel, probabilistic approaches have emerged to address model uncertainty in real-world deployment. Celikkan et al. (2023) proposed a Bayesian DeepLabv3 framework for crop–weed segmentation, extending deterministic CNNs with uncertainty quantification. Their model achieved competitive results on the Sugarbeets2016 dataset (mIoU 88.37%) and PhenoBench UAV dataset (mIoU 85.77), while also providing predictive entropy and mutual information maps to identify areas of high uncertainty, such as crop–weed overlaps. This probabilistic modeling not only improves reliability but also enables safer integration into automated weeding systems by flagging uncertain predictions where misclassification could lead to costly errors [9].
Most existing agricultural semantic-segmentation studies remain binary (vegetation vs. soil or crop vs. weed), with only a limited number extending to genuine multi-species classification. The few multiclass examples—such as VGG16 networks segmenting barley and oilseed radish or U-Nets mapping alpine meadow species—illustrate the potential but remain exceptions. Consequently, our focus on detailed multiclass vegetation cover estimation represents a relatively novel direction in this field.

2.4 Encoders

The encoder is the component responsible for extracting increasingly abstract and high-level features from the input image. Its primary function is to transform raw pixel data into a compact representation that captures the essential structures and patterns necessary for the downstream task of semantic segmentation. The encoder typically consists of a sequence of convolutional layers, activation functions, and pooling operations. At the initial stages, convolutional filters detect low-level features such as edges, textures, and simple shapes. As the network deepens, successive layers capture progressively more complex structures, including object parts and semantic categories. This allow the network to achieve two goals:

- **Hierarchical abstraction**: by stacking layers, the encoder learns feature hierarchies ranging from local details to global context.
- **Dimensionality reduction**: downsampling compresses spatial information, reducing computational cost and allowing the network to focus on salient regions of the image.

The output of the encoder is a dense, lower-dimensional latent representation of the input image. This representation retains the semantic information necessary for segmentation, while discarding redundant details such as small variations in illumination or noise. The latent features are then passed to the decoder, which reconstructs the pixel-level segmentation map from this compressed representation. Encoders can be either custom-designed for specific tasks or pre-trained backbones originally developed for image classification. In semantic segmentation, using pre-trained models such as ResNet, VGG, MobileNet, or EfficientNet as encoders is a common strategy. These backbones leverage transfer learning, bringing general visual knowledge learned from large-scale datasets (e.g., ImageNet [15]) into agricultural applications, thus improving accuracy and reducing training time. In this work, two popular encoder architectures are investigated: **MobileNetV2**, and **EfficientNet-B0**.

2.4.1 MobileNetV2

MobileNetV2, introduced in 2019 [66] builds upon the foundation of MobileNetV1 [27] by retaining its simplicity and requiring no specialized operators, while significantly improving accuracy for various image classification and detection tasks designed for mobile applications. The key innovation in MobileNetV2 is the introduction of the inverted residual with linear bottleneck. This module processes input through three stages:

1. It first expands the low-dimensional compressed representation to a higher dimension.
2. It filters this high-dimensional representation using a lightweight depthwise convolution.
3. Finally, it projects the features back to a low dimensional space using a linear convolution.

For an input set of real images, the set of layer activations (for any layer L_i) forms a "manifold of interest". There are two properties that are indicative of the requirement that the manifold of interest should lie in a low-dimensional subspace of the higher-dimensional activation space: if the manifold of interest remains non-zero volume after ReLU transformation, it corresponds to a linear transformation

and ReLU is capable of preserving complete information about the input manifold, but only if the input manifold lies in a low-dimensional subspace of the input space. These two insights provide an empirical hint for optimizing the neural architectures: assuming the manifold of interest is low-dimensional it can be captured by inserting linear bottleneck layers into the convolutional blocks. Experiments show that using linear layers is crucial as it prevents non linearities from destroying too much information while using non-linear layers indeed hurts the performance by several percent. Figure 2.3 provides a schematic visualization of the difference between the residual block and the inverted residual block. The motivation for inserting shortcuts is similar to that of classical residual connections: improve the ability of a gradient to propagate across multiplier layers.

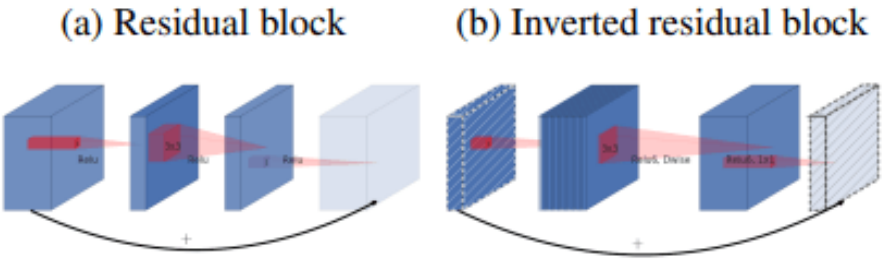


Figure 2.3: The difference between residual block and inverted residual. Image taken from [66].

One interesting property of the architecture is that it provides a natural separation between the input/output domains of the building blocks (bottleneck layers), and the layer transformation— that is a non-linear function that converts input to the output. The former can be seen as the capacity of the network at each layer, whereas the latter as the expressiveness. The MobileNetV2 architecture contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. It uses ReLU6 as the non-linearity and a kernel size 3×3. With the exception of the first layer, it uses constant expansion rate throughout the network (expansion rates between 5 and 10 result in nearly identical performance curves, with smaller networks being better off with slightly smaller expansion rates and larger networks having slightly better performance with larger expansion rates). Moreover the inverted residual bottleneck layers allow a particularly memory efficient implementation which is very important for mobile applications. In those layers the total amount of memory would is dominated by the size of bottleneck tensors, rather than the size of tensors that are internal to bottleneck (which number is way larger). Figure 2.4 illustrates the structure of MobileNetV2.

2.4.2 EfficientNet-B0

EfficientNet-B0, introduced by Tan and Le in 2019 [71], is the baseline architecture of the EfficientNet family of convolutional neural networks designed to optimize both accuracy and efficiency through a systematic model scaling method. Traditional approaches to scaling neural networks typically increase either the depth (number of layers), width (number of channels), or input resolution independently, often leading to suboptimal results. EfficientNet introduces compound scaling, a principled method that uniformly scales these three dimensions using a set of fixed coefficients. This approach yields models that achieve state-of-the-art accuracy while requiring significantly fewer parameters and floating-point

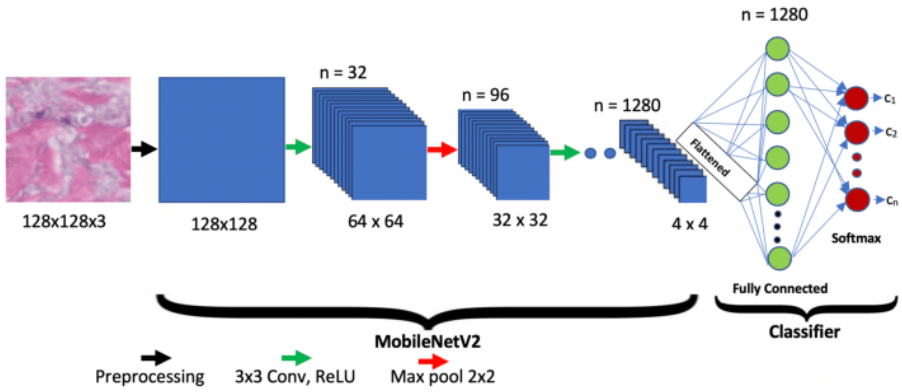


Figure 2.4: MobileNetV2 architecture overview.

operations compared to conventional architectures. The baseline network, EfficientNet-B0, was discovered through neural architecture search [70] with accuracy and efficiency constraints. Its building block is the mobile inverted bottleneck convolution (MBConv), originally introduced in MobileNetV2. Each MBConv block expands the input tensor to a higher-dimensional space, applies depthwise separable convolutions for efficient feature extraction, and then projects the features back to a lower dimension. To further improve representational power, EfficientNet integrates Squeeze-and-Excitation (SE) blocks into the MBConv structure. SE blocks adaptively recalibrate channel-wise feature responses by modeling interdependencies between channels, allowing the network to emphasize informative features while suppressing less useful ones.
Formally, the compound scaling method can be expressed as:

$$depth = \alpha^{\phi}, \quad width = \beta^{\phi}, \quad resolution = \gamma^{\phi}$$

subject to the constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ where ϕ s a user-specified coefficient controlling overall model size. The baseline EfficientNet-B0 corresponds to $\phi = 1$ and larger variants (B1–B7) are obtained by increasing ϕ. The EfficientNet-B0 architecture begins with a standard convolutional layer, followed by a sequence of MBConv blocks with varying expansion ratios, kernel sizes, and number of filters. Specifically, it uses expansion ratios of 1, 6, or higher depending on the stage, and kernel sizes of 3×3 or 5×5 to balance receptive field and computational cost. The network concludes with a convolutional head, global average pooling, and a fully connected classification layer. Figure 2.5 provides an overview of the EfficientNet-B0 architecture. An advantage of EfficientNet-B0 is its ability to achieve high accuracy with a compact parameter budget, making it suitable as an encoder backbone for semantic segmentation tasks in resource-constrained environments such as precision agriculture. Its combination of MBConv, SE blocks, and compound scaling provides a favorable trade-off between model capacity and efficiency, ensuring portability to mobile and embedded devices while maintaining competitive performance.

2.5 Decoders

The decoder is the counterpart to the encoder in semantic segmentation architectures. While the encoder progressively reduces the spatial resolution of the input image to capture abstract semantic

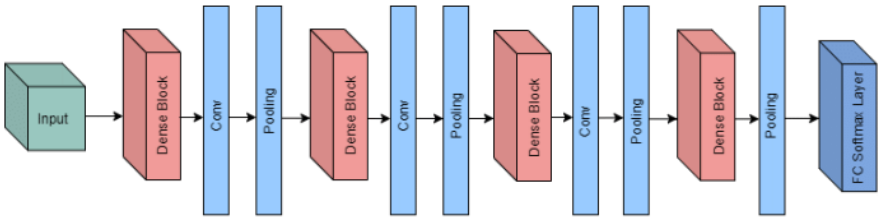


Figure 2.5: EfficientNet-B0 architecture overview.

features, the decoder restores this information into a full-resolution segmentation map. Its main role is to combine semantic richness with spatial precision, enabling pixel-level classification. A decoder typically employs upsampling operations, transposed convolutions, or interpolation strategies to increase the spatial dimensions of the feature maps. To recover fine-grained details lost during downsampling, skip connections are often introduced between encoder and decoder layers. These links reintroduce low-level spatial information that is necessary for accurate boundary delineation.
Overall, the decoder transforms the compressed latent representation produced by the encoder into dense predictions, striking a balance between localization accuracy and semantic understanding. In this work, three popular decoder architectures are investigated: **U-Net**, **U-Net++**, and **DeepLabV3**.

2.5.1 UNet

U-Net, introduced by Ronneberger et al. in 2015 [64], is one of the most widely adopted decoder architectures for semantic segmentation, particularly in biomedical and agricultural imaging tasks. Its characteristic "U" shape (as depicted in Figure 2.6) consists of a contracting path (encoder) and an expansive path (decoder) connected by symmetric skip connections. The decoder progressively upsamples feature maps using transposed convolutions, doubling spatial resolution at each stage. At each upsampling step, feature maps from the encoder are concatenated with the decoder's activations, providing localized spatial information to refine predictions. This design enables U-Net to capture both contextual information (from the encoder) and fine details (from the skip connections). Formally, given encoder feature maps $f_{\text{dec}}^{(i)}$ at stage i, the decoder stage computes:

$$f_{\text{dec}}^{i} = \text{UpConv}\big(f_{\text{dec}}^{i+1}\big) \oplus f_{\text{enc}}^{i}$$

where $\oplus$ denotes concatenation. U-Net's simplicity and efficiency have made it a backbone for many domain-specific segmentation tasks, including precision agriculture, where it achieves reliable crop–weed distinction.

2.5.2 UNet++

U-Net++, proposed by Zhou et al. in 2018 [77], extends the original U-Net by redesigning the skip connections between encoder and decoder. Instead of direct concatenation, U-Net++ introduces nested and dense skip pathways composed of intermediate convolutional blocks. This redesign aims to reduce the semantic gap between encoder and decoder feature maps. By progressively refining encoder features before merging them into the decoder, U-Net++ enhances segmentation accuracy, particularly for small

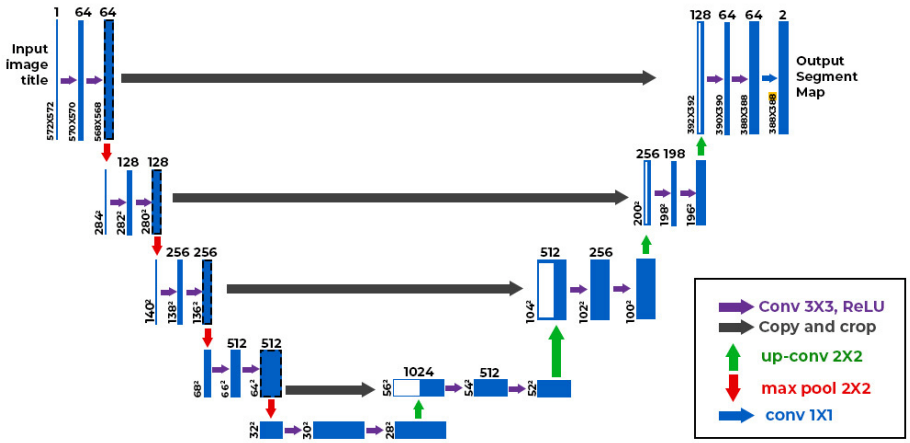


Figure 2.6: UNet architecture overview.

or irregular objects. The architecture can be expressed as a set of intermediate nodes $X_{i,j}$ where i denotes the depth of the encoder and j the stage of refinement:

$$X_{i,j} = H(X_{i,j-1}) \oplus \bigoplus_{k=1}^{i-1} X_{k,j}$$

with $H(\cdot)$ representing a convolutional block. This dense connectivity results in a deeper supervision scheme, improving gradient flow and allowing flexible pruning for different computational budgets. In agricultural segmentation, U-Net++ often outperforms U-Net in capturing fine-grained boundaries, such as distinguishing overlapping leaves.

2.5.3 DeepLabV3

DeepLabV3, introduced in 2017 by Chen et al. [10], represents a shift from traditional symmetric encoder–decoder designs. Instead of explicit upsampling with skip connections, it focuses on capturing multi-scale contextual information through Atrous Spatial Pyramid Pooling (ASPP). ASPP applies parallel atrous (dilated) convolutions with different dilation rates, enabling the network to capture features at multiple receptive field sizes without increasing computational cost. This approach is especially useful for handling objects that appear at varying scales in agricultural imagery. The output of ASPP can be formalized as:

$$F_{\text{ASPP}} = \bigoplus_{r \in R} \text{Conv}_{\text{dilation}=r}(F_{\text{enc}})$$

where R is the set of dilation rates. Finally, DeepLabV3 upsamples its predictions to the original image resolution, often using bilinear interpolation instead of learned transposed convolutions. Although it lacks explicit symmetric skip connections, the strong contextual representation makes it highly effective for semantic segmentation. In agricultural applications, DeepLabV3 is well-suited for segmenting heterogeneous fields, where both small weeds and large crop structures coexist.

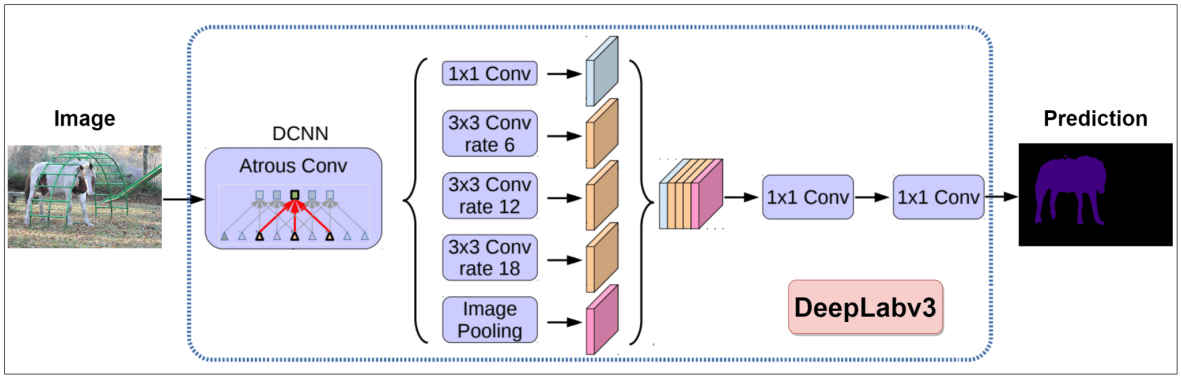


Figure 2.7: DeepLabV3 architecture overview.

3

Dataset

A fundamental requirement for training and evaluating deep learning models in semantic segmentation is the availability of high-quality datasets. In the context of precision agriculture, datasets play a role in enabling models to learn the fine-grained differences between crops, weeds, and soil under diverse and challenging field conditions. Unlike general-purpose computer vision, where large benchmark datasets such as ImageNet or COCO [43] have driven rapid progress, agricultural research still faces limitations due to the scarcity and heterogeneity of annotated data.

This chapter provides an overview of the datasets considered in this thesis. We begin with a survey of publicly available agricultural datasets, drawing on recent work that catalogs and organizes field-based image collections suitable for computer vision tasks. The survey highlights the main characteristics, strengths, and limitations of existing datasets, with a particular focus on those addressing crop–weed segmentation. Subsequently, we describe in detail the specific datasets selected for the experiments in this work, namely **WE3DS** [36] and **CropAndWeed** [68], outlining their structure, annotation types, and relevance to the research objectives. Finally, we introduce an additional dataset consisting of images collected in the gardens of the University of Udine. This dataset is used exclusively for post-processing evaluation and is not involved in the training phase.

3.1 Agriculture dataset survey

The rapid development of deep learning has led to significant progress in computer vision (CV) for agriculture, with applications ranging from precision weeding to disease monitoring and yield estimation. However, unlike general-purpose computer vision, where large-scale datasets such as ImageNet or CIFAR [37] have enabled breakthroughs, agricultural CV research still suffers from a relative scarcity of openly available, high-quality datasets. This gap is caused by several factors, including the high variability of agricultural environments, the complexity of collecting and annotating data in natural field conditions, and the reluctance of some researchers to release datasets. To compensate for the lack of open datasets, Heider et al. (2025) [25] conducted a comprehensive survey of agricultural field datasets and created an openly accessible online catalog [1]. The catalog compiles 45 publicly available RGB image datasets, representing natural field scenes captured under realistic conditions, and provides metadata

[1] Available at https://smartfarminglab.github.io/field_dataset_survey/

such as annotation type, crop species, dataset size, and intended task.
The datasets were identified through an extensive screening of the Computers and Electronics in Agriculture journal, the arXiv preprint server, Google Scholar, and repositories such as Kaggle and Zenodo. From an initial pool of 136 datasets, 45 were selected according to three criteria:

1. **Domain coherence**: images must depict plants or weeds in natural field conditions, excluding synthetic or artificially staged environments.

2. **Annotation quality**: datasets must provide sufficient and well-structured ground truth labels.

3. **Image quality**: datasets must feature consistent resolution, adequate lighting, and minimal noise.

The selected datasets span several agricultural computer vision tasks:

- Weed detection and classification (29 datasets): e.g., *DeepWeeds* (Olsen et al., 2019 [54]), *Global Wheat Head Detection (GWHD)* (David et al., 2021 [13]), *YOLOWeeds* (Dang et al., 2023 [11]).

- Plant disease and pest detection (9 datasets): e.g., *RiceLeafDiseases* (Antony et. al., 2023 [45]), *SoyNet* (Rajput et. al., 2023[60]), *PaddyDoctor* (Petchiammal et. al., 2023 [56]).

- Seedling and crop detection/classification (6 datasets): e.g., *PlantSeedlingClassification* (Giselsson et al., 2017 [22]), *DeepSeedling* (Jiang et al., 2019 [30]).

- Crop segmentation, phenotyping, and counting: e.g., *PhenoBench* (Weyler et al., 2024 [74]), *SoybeanNet* (Li et al., 2024 [40]), *MFWD* (Genze et al., 2024 [20]).

Collectively, these datasets account for more than 2.5 terabytes of image data, with variations ranging from small, high-quality collections with detailed bounding box annotations, to massive repositories with millions of labeled instances. While the catalog helps close the quantitative gap in open datasets, significant inter-dataset variability and the task-specific design of many collections continue to constrain their universal applicability.

3.1.1 WE3DS dataset

The WE3DS dataset (Weed and crop species segmentation) [36] is a publicly available benchmark designed to support research on semantic segmentation for precision agriculture. It specifically addresses the challenge of distinguishing between multiple plant species, weeds, and background soil in real agricultural environments. Unlike binary crop–weed classification datasets, WE3DS provides multiclass segmentation, enabling pixel-wise categorization across diverse species. The development of WE3DS was motivated by the limitations of prior agricultural datasets. Most existing resources only distinguish between two or three classes (soil, crop, weed), which is insufficient for species-specific vegetation cover estimation. Furthermore, many datasets were captured in controlled conditions with artificial lighting or opaque shrouds to eliminate variability, which restricts their applicability in real-world scenarios. WE3DS overcomes these constraints by providing RGB-D images collected under natural outdoor lighting conditions, capturing realistic challenges such as variable illumination (sunny, cloudy, and mixed weather conditions across multiple measurement dates), occlusions and growth stages (overlapping plants

and morphological variations across early phenological stages), soil diversity and background noise (including residues, clods, and uneven surfaces).

A distinguishing feature of WE3DS is the inclusion of distance (depth) information, calculated via stereo vision. The disparity map, derived from normalized cross-correlation matching, was converted into distance measurements in 10^{-1} mm units. On average, only 0.72% of pixels per image lacked valid depth data, ensuring reliable 3D information for nearly the entire scene. This additional modality allows for multimodal semantic segmentation, combining color and spatial cues, which has been shown to improve performance by up to 6–7% mIoU compared to RGB-only models.

Dataset composition

The WE3DS dataset consists of 2,568 RGB-D image pairs, each comprising an RGB image and its corresponding distance map. The original images were captured at a resolution of 1600 × 1144 pixels. Data collection took place over two consecutive growing seasons (2020 and 2021) at the experimental farm of the University of Natural Resources and Life Sciences (BOKU) in Groß-Enzersdorf, Austria. The image acquisition system was a custom-built stereo camera setup composed of two XIMEA MC023CG-SY industrial cameras equipped with global shutters and mounted with a baseline of 4–5 cm. This configuration provided a ground resolution of approximately 0.4 mm per pixel and a depth accuracy of about 1.6 mm when operated at a working height of 90 cm. The dataset covers a wide range of plant species and includes a total of 18 semantic classes: soil as background, 7 crop species, and 10 weed species. Additionally, a void class is used for uncertain or unknown plant types. In total, 4038 crop instances and 7506 weed instances were annotated, with each image containing an average of 4.5 plants, distributed between crops and weeds. The complete list is represented in Table 3.1 while Figure 3.1 shows the distribution of annotated pixels per class across all images.

Index	Class name	Crop/Weed
0	Void	
1	Soil	
2	Broad bean	Crop
3	Corn spurry	Weed
4	Red-root amaranth	Weed
5	Common buckwheat	Crop
6	Pea	Crop
7	Red fingergrass	Weed
8	Common wild oat	Weed
9	Cornflower	Weed
10	Corn cockle	Weed
11	Corn	Crop
12	Milk thistle	Weed
13	Rye brome	Weed
14	Soybean	Crop
15	Sunflower	Crop
16	Narrow-leaved plantain	Weed
17	Small-flower geranium	Weed
18	Sugar beet	Crop

Table 3.1: WE3DS dataset composition.

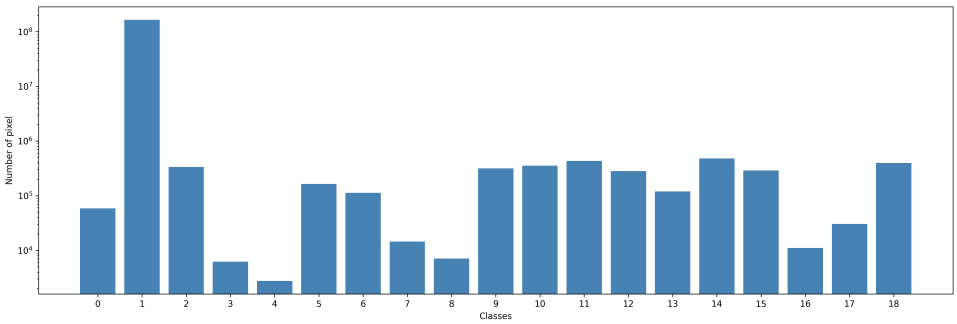


Figure 3.1: Distribution of annotated pixels per class across all WE3DS images.

Annotation process

The WE3DS dataset provides annotations in the form of pixel-level semantic masks. Each image was manually labeled using the CVAT tool, where annotators drew polygons around individual plant instances. These polygons were then converted into dense segmentation masks, with each pixel assigned to one of the predefined classes (soil, 17 plant species, or void). Pixel-level masks are particularly important in agricultural applications, because plant structures such as leaves and stems have irregular shapes and overlapping regions that cannot be accurately captured by bounding boxes. Consequently, the WE3DS annotations allow for fine-grained analysis of plant distribution and morphology, which supports more precise tasks such as selective weeding or biomass estimation. Figure 3.2 shows an example of an annotation present in the dataset.

3.1.2 CropAndWeed dataset

The CropAndWeed dataset (Steininger et al., 2023 [68]) was proposed to overcome the data scarcity problem in precision agriculture by providing a large-scale, diverse, and richly annotated benchmark specifically designed for crop–weed discrimination. It integrates a multi-modal annotation scheme and a broad representation of plant species in early growth stages. With more than 8,000 high-resolution images and approximately 112,000 annotated instances distributed across 74 plant classes (16 crops and 58 weeds), it represents one of the most comprehensive resources available in this research domain. The dataset was designed with two key objectives: to capture the high variability of agricultural environments, including lighting, soil texture, and plant distribution, and to provide annotations suitable for multiple learning tasks beyond simple classification, such as object detection, semantic segmentation, stem localization, and panoptic segmentation. By focusing on early plant growth stages the dataset supports applications that target selective herbicide spraying or mechanical removal before weeds can significantly reduce crop yield.

Dataset composition

Data were collected over a four-year period between March and July, with particular attention to variability in daytime, season, weather, and soil conditions. Images were acquired using a semi-professional SLR camera with a full-frame sensor and a fixed focal length of 50 mm, from a top-down perspective at approximately 1.1 meters. Each acquisition session corresponded to a unique site or experimental

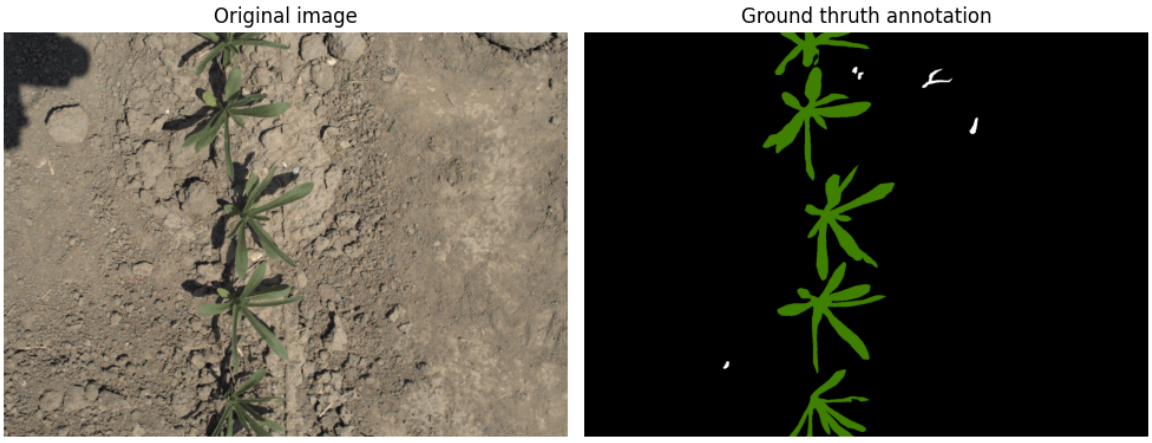


Figure 3.2: An example image from the WE3DS dataset along with its corresponding segmentation masks.

plot and typically included around 20 images taken at regular intervals to minimize redundancy. This approach produced a dataset that captures a wide spectrum of variability. Differences in lighting (from overcast to strong sunlight), soil moisture (dry, cracked soil to wet, reflective surfaces), and background clutter (rocks, straw, fertilizer, trash) were deliberately included. In addition, negative samples containing no vegetation were recorded to improve robustness in detection and segmentation tasks.
The dataset is divided in two complementary subsets:

- **Application Set**: this set reflects the complexity of natural environments, including heterogeneous soil, mixed plant species, and uncontrolled illumination conditions. It is therefore particularly valuable for realistic validation and testing.
- **Experimental Set**: specifically cultivated in controlled outdoor plots to enrich the dataset with underrepresented species. By planting selected crops and weeds in isolation or in defined combinations, the authors ensured clear visual identification even at early stages and balanced representation of rare weeds, which are usually scarce in natural field collections.

In total, the CropAndWeed dataset comprises 43,814 recorded images, of which 8,034 were selected for fine-grained annotation, collected across 929 recording sessions that span both experimental cultivation plots and real-world agricultural fields. These annotated images contain approximately 111,953 individual plant instances, distributed over a total of 74 classes, including 16 crop species and 58 weed species. The dataset is characterized by a naturally imbalanced distribution, as weeds are generally more frequent yet visually smaller than crops, reflecting their occurrence in real agricultural environments. To address this imbalance, the creators specifically cultivated underrepresented weed species in controlled outdoor plots, ensuring sufficient representation even for rare categories. Additionally, a fallback class named Vegetation was introduced to group instances that are either too small (less than 162 pixels in bounding box area) or visually ambiguous, ensuring consistent treatment during annotation and evaluation. Figure 3.3 shows the distribution of annotated pixels per class across all images while the table

at the end of the chapter shows the full list of annotated species included in the CropAndWeed dataset.

Annotation process

CropAndWeed stands out for its multi-modal annotation scheme, which supports a wide range of machine learning tasks. Each plant instance is annotated with:

- Bounding boxes for object detection.
- Semantic segmentation masks for pixel-wise classification.
- Stem positions as anchor points, enabling precise localization for robotic manipulation tasks.

The annotation process followed a semi-automated pipeline. Initial vegetation masks were generated using color-based thresholding and later refined with preliminary CNN-based segmentation models trained on the dataset itself. Human annotators then manually corrected these pre-segmentations, adjusted bounding boxes, and labeled stem positions. To ensure consistency, ambiguous or dense samples were reviewed by multiple annotators in a voting process. The dataset also includes meta-annotations: GPS coordinates, timestamps, camera parameters, and environmental descriptors (lighting conditions, soil moisture, soil type). These enrichments allow researchers to study not only plant recognition but also model robustness under varying field conditions. Figure 3.4 shows an example of an annotation present in the dataset.

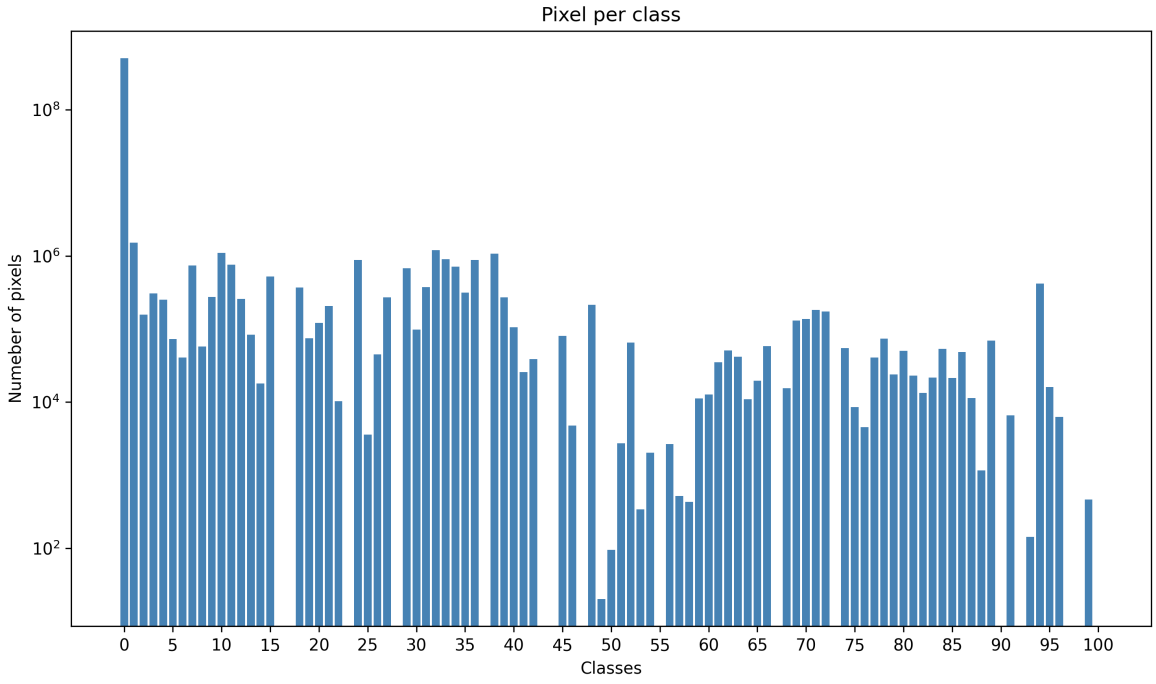


Figure 3.3: Distribution of annotated pixels per class across all images.

3.2 Post-Evaluation dataset

This section describes an additional dataset consisting of images collected in the gardens of the University of Udine. Unlike the datasets presented in Section 3.1, this dataset is not used for training or validation.

Species	Botanical Classification	Crop/Weed	Sessions	Images	Instances	Mapping
Maize	Zeamays	C	167	1834	6018	1
Sugar beet	Beta vulgaris subsp. vulgaris	C	133	1210	6161	2
Pea	Pisum sativum	C	18	176	621	6
Courgette	Cucurbita pepo subsp. p. c. g.	C	1	18	22	-
Pumpkins	Cucurbita	C	11	317	615	8
Potato	Solanum tuberosum	C	15	108	371	5
Flat leaf parsley	Petroselinum stivum	C	1	14	35	-
Curly leaf parsley	Petroselium crispum	C	1	10	35	-
Cowslip	Primula veris	C	1	19	108	-
Poppy	Papaver	C	1	2	25	22
Sunflower	Helianthus annuus	C	62	590	1879	4
Sage	Salvia	C	1	2	23	-
Common bean	Phaseolus vulgaris	C	2	13	109	7
Faba bean	Vicia faba	C	20	216	952	7
Hybrid goosefoot	Chenopodiastrum hybridum	W	72	205	329	11
Black-bindweed	Fallopia convolvulus	W	15	45	195	12
Cockspur grass	Echinochloa crus-galli	W	118	270	1280	10
Red-root amaranth	Amaranthus retroflexus	W	203	491	3484	10
White goosefoot	Chenopodium album	W	322	1121	3180	11
Thorn apple	Datura stramonium	W	46	167	469	15
Potato weed	Galinsoga parviflora	W	219	636	2563	16
German chamomile	Matricaria chamomilla	W	102	300	2113	17
Creeping thistle	Cirsium arvense	W	86	410	2693	18
Field milkt histle	Sonchus arvensis	W	202	513	2623	18
Purslane	Portulaca oleracea	W	22	30	70	-
Black nightshade	Solanum nigrum	W	9	9	14	15
Mercuries	Mercurialis	W	253	812	1147	19
Geraniums	Geranium	W	41	90	667	20
Cleavers	Galium aparine	W	8	20	35	-
Meadow-grass	Poa	W	232	763	3785	10
Frosted orach	Atriplex laciniata	W	1	1	1	11
Black horehound	Ballota nigra	W	1	1	1	24
Shepherd' spurse	Capsella bursa-pastoris	W	4	5	8	21
Field bindweed	Convolvulus arvensis	W	57	161	533	15
Common mugwort	Artemisia vulgaris	W	2	2	2	-
Hedge mustard	Sisymbrium	W	19	22	29	21
Speedwell	Veronica	W	37	53	108	23
Broadleaf plantain	Plantago major	W	9	12	12	23
Whiteball-mustard	Calepina irregularis	W	1	1	2	21
Peppermint	Mentha piperita	W	18	84	602	24
Field pennycress	Thlaspi arvense	W	22	79	118	21
Corn spurry	Spergula arvensis	W	19	125	922	13
Purple crabgrass	Digitaria sanguinalis	W	9	81	473	10
Common fumitory	Fumaria officinalis	W	11	73	227	22
Ivy-leaved speedwell	Veronica hederifolia	W	10	60	237	23
Annual meadow grass	Poa annua	W	6	24	447	10
Redshank	Persicaria maculosa	W	30	129	428	12
Rough meadow-grass	Poa trivialis	W	4	17	405	10
Green bristlegrass	Setaria viridis	W	287	1648	12132	10
Small geranium	Geranium pusillum	W	42	406	4428	20
Cornflower	Cyanus segetum	W	27	270	1780	18
Common corn-cockle	Agrostemma githago	W	20	211	1601	18
Wall barley	Hordeum murinum	W	16	88	466	10
Annual fescue	Festuca myuros	W	4	9	85	10
Purple dead-nettle	Lamium purpureum	W	27	39	47	24
Ribwort plantain	Plantago lanceolata	W	14	100	943	23
Pineappleweed	Matricaria discoidea	W	25	69	608	17
Common chickweed	Stellaria media	W	36	86	219	14
Hedge mustard	Sisymbrium officinale	W	25	121	917	21
Soft brome	Bromus hordeaceus	W	8	47	368	10
Wild pansy	Viola tricolor	W	46	96	263	-
Yellow rocket	Barbarea vulgaris	W	19	81	379	21
Common wild oat	Avena fatua	W	22	105	390	10
Red poppy	Papaver rhoeas	W	15	39	630	22
Rye brome	Bromus secalinus	W	8	66	594	10
Knotgrass	Polygonum aviculare agg.	W	43	67	408	12
Prickly lettuce	Lactuca serriola	W	3	5	5	18
Copse-bindweed	Fallopia dumetorum	W	194	858	1790	12
Common buckwheat	Fagopyrum esculentum	W	19	83	211	12
Garlic	Allium sativum	C	1	1	2	-
Soybean	Glycine max	C	42	336	4039	3
Wild carrot	Daucus carota supsp. carota	W	18	42	69	-
Field mustard	Sinapis arvensis	W	28	49	402	21
Common dandelion	Taraxacum sect. Ruderalia	W	1	1	1	-
Vegetation	Vegetation	W	788	5066	33000	26
Overall			**929**	**8034**	**111953**	

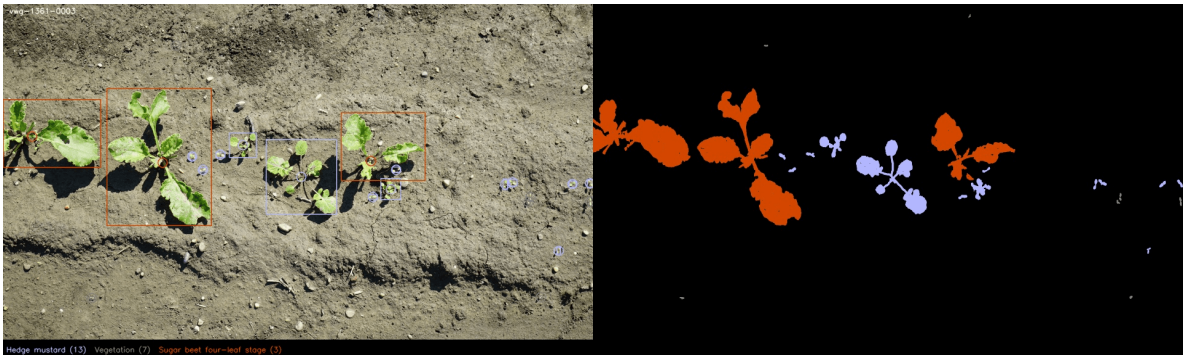


Figure 3.4: An example image from the CropAndWeed dataset along with its corresponding segmentation masks.

Instead, it serves exclusively for post-processing evaluation, allowing us to assess the generalization of the trained models on a separate set of real-world images acquired under uncontrolled conditions.
The dataset used in this work derives from a controlled intercropping experiment designed to investigate observer-related variability in vegetation cover estimation. The trial featured lentil and buckwheat grown under five different cropping systems:

- Lentil only.
- Within row intercropping with 25% buckwheat seed rate (MIX25).
- Within row intercropping with 50% buckwheat seed rate (MIX50).
- Alternate row intercropping (ALT).
- Buckwheat only.

Crops were sown on April 9th 2024, and two vegetation surveys were conducted at distinct growth stages: **T1** which contains images of 15 days after emergence (third true leaf stage of lentil, second true leaf stage of buckwheat) and **T2** which contains images of 38 days after emergence (second branch stage of lentil, full flowering stage of buckwheat). The presence of vegetation (buckwheat, lentil, and weeds) in each field plot was independently assessed by two human observers (one expert and one novice) in two 0.35 m^2 sampling areas, delimited by a 0.5 m $\times$ 0.7 m frame. As a result, eight observations (on corresponding sampling areas) were conducted for each cropping system at each sampling date.
Each sampling area was also photographed from a fixed height of 1 meter, maintaining a consistent direction and orientation of the device (Samsung Galaxy A42 smartphone). The pictures were then cropped and adjusted to match the dimensions of the sampling frame, an example of the resulting pictures is shown in Figure 3.5.

Dataset composition

The final result of this process was a set of 80 image-mask pairs, where each RGB image is paired with the corresponding segmentation mask that encodes the four semantic classes (3 plant classes and the background). All the images in the dataset (and the corresponding masks) are high-resolution. As in the previous cases, also this dataset is highly imbalanced, with certain classes (particularly weeds)

Figure 3.5: Example of pictures included in the dataset.

occupying only a small fraction of the area of the image in many samples. This imbalance is slightly reduced at T2, but still present. Figure 3.6 shows the distribution of annotated pixels per class across all samples at T1 and T2, respectively.

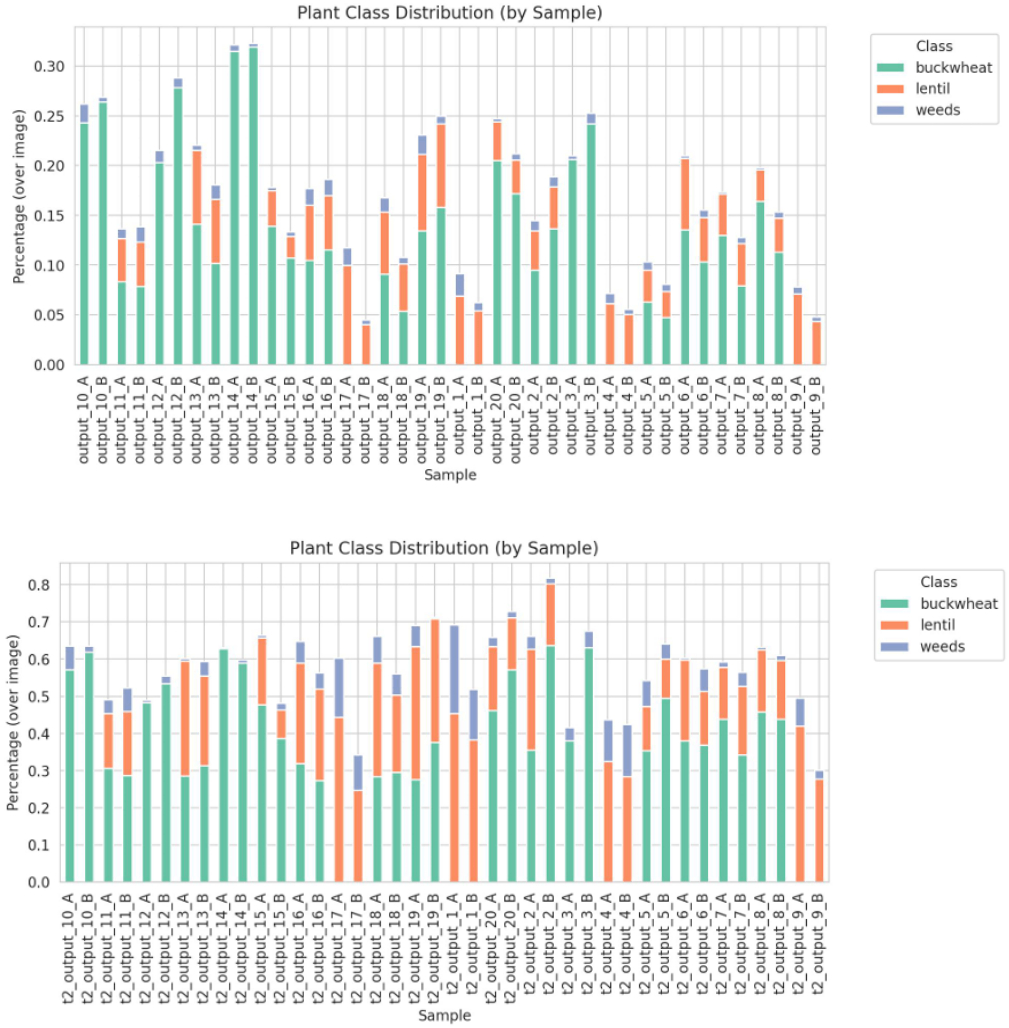


Figure 3.6: Distribution of annotated pixels per class across all samples at T1 and T2.

Annotation process

The 80 images collected during the experiment were annotated to generate high quality masks for each vegetation category. More precisely, during the annotation process each pixel in the images was assigned to one of four (mutually exclusive) classes: *buckwheat*, *lentil*, *weeds*, *background*. All the annotations were done using a semi-automatic tool that combines manual (human) input and AI assisted segmentation.

The AI assisted components of the tool are powered by the Segment Anything Model (SAM) [35] for the general purpose segmentation, and VegAnn [46], a model trained to distinguish vegetation from background (binary segmentation task) that is useful to refine the masks generated by SAM. The annotations were carried out by the novice observer, who was previously trained in both field estimation techniques and the use of the annotation tool. The reliability of the annotations is supported by their consistency with the expert's field observations. During the vegetation surveys, both the novice and expert observers independently assessed vegetation cover in the same sampling areas, and their estimates showed strong agreement. This cross-validation ensures that the image annotations, while generated by a single annotator using a semi-automated tool, are aligned with expert-level assessments, supporting the overall trustworthiness of the dataset. Figure 3.7 shows an example image from the dataset (taken at T1) along with its corresponding segmentation masks. The yellow part denotes the buckwheat, the red one the lentils and the blue one the weeds.

Figure 3.7: An example image from the post-evaluation dataset along with its corresponding segmentation mask.

4

Methodology

This chapter presents the methodological framework adopted in this work. It begins with a clear statement of the problem addressed, followed by the description of the preprocessing steps applied to the datasets used in training and evaluation. The subsequent sections introduce the combinations of encoder–decoder models investigated, together with the selection of hyperparameters, loss functions, and optimization strategies. Then, the experimental setup is detailed in terms of batch size and training configuration. Finally, a dedicated section describes the evaluation method, including the metrics and criteria adopted to assess segmentation performance. Overall, the chapter provides a complete overview of the pipeline implemented for semantic segmentation, from raw data preparation to model training and evaluation.

4.1 Problem statement

The primary objective of this work is to develop a model capable of performing pixel-wise semantic segmentation for estimating vegetation cover across diverse cropping systems. Formally, the task can be defined as follows: given an input RGB image I represented as a tensor with shape $3 \times H \times W$ captured from a top down perspective, the model must generate a corresponding segmentation mask $M \in \{0, 1, ..., C-1\}^{H \times W}$. In this mask, each pixel is assigned to one of C semantic classes. The problem consists also in ensuring that the model not only performs accurately on the training data but also generalizes effectively across variations in cropping systems, plant growth stages, planting densities, spatial arrangements, and visual occlusions. To further assess this capability, we also employ a post-evaluation dataset, which provides an additional benchmark for testing the model's effectiveness in scenarios not encountered during training, thereby offering a more rigorous measure of generalization and practical applicability.

4.2 Data preparation

In this section, we outline the preprocessing steps applied to the raw dataset prior to model training. To maintain consistency and reliability, all data underwent a standardized preprocessing pipeline. First, the dataset image identifiers were collected and merged into a single list. The dataset was then randomly

divided into three disjoint subsets: 70% for training, 15% for validation, and 15% for testing. This stratification ensures that the model is trained on the majority of the available data while preserving sufficient samples for both hyperparameter tuning and unbiased performance evaluation. A custom PyTorch Dataset class was implemented to handle data loading. For each sample, the class retrieves the corresponding RGB image and its ground truth segmentation mask, both identified by a common image ID. Images were read using OpenCV [6] and converted to RGB format, while masks were loaded in grayscale mode. To ensure pixel-wise alignment between images and masks, the same transformations were consistently applied to both. All images and masks were resized to a fixed resolution of 256 $\times$ 256 pixels, balancing the trade-off between computational efficiency and preservation of spatial detail. The transformations were implemented using the Albumentations library [8], which allows consistent preprocessing across the training, validation, and test sets. After resizing, images were converted into PyTorch tensors and scaled to the range $[0, 1]$ by dividing pixel values by 255, while masks were converted into integer tensors representing class labels. Finally, the preprocessed datasets were wrapped into PyTorch DataLoaders. During training, data were fed in mini-batches (a deeper understanding of the batch size will be discussed in the Section 4.4.3), with the training loader using shuffling to improve model generalization, while the validation and test loaders preserved a fixed order to ensure reproducibility of results.

4.3 Models considered

We evaluate multiple combinations of segmentation Encoder-Decoder architectures to identify effective solutions for the semantic segmentation of vegetation, systematically comparing the CNN-based architectures.

Encoder	Decoder	Encoder size	Decoder size	Total size	GFLOPs	Inference Time
EfficientNet-B0	UNet	4.01M	2.24M	6.25M	10.81	8.17 ms/img
MobileNetV2	UNet	2.22M	4.40M	6.63M	14.26	4.68 ms/img
EfficientNet-B0	UNet++	4.01M	2.24M	6.25M	21.11	9.49 ms/img
MobileNetV2	UNet++	2.22M	4.60M	6.83M	18.68	6.13 ms/img
EfficientNet-B0	DeepLabV3	4.01M	3.30M	7.31M	13.65	10.04 ms/img
MobileNetV2	DeepLabV3	2.22M	10.42M	12.65M	51.14	7.16 ms/img

Table 4.1: Summary of all the Encoder-Decoder combinations tested along with their parameters count (in millions) broken down into encoder, decoder, total model size, GFLOPs and Inference time for the WE3DS dataset.

Encoder	Decoder	Encoder size	Decoder size	Total size	GFLOPs	Inference Time
EfficientNet-B0	UNet	4.04M	2.24M	6.28M	13.91	8.28 ms/img
MobileNetV2	UNet	2.25M	4.40M	6.66M	17.36	4.83 ms/img
EfficientNet-B0	UNet++	4.04M	2.56M	6.60M	24.20	9.63 ms/img
MobileNetV2	UNet++	2.25M	4.60M	6.86M	21.78	6.44 ms/img
EfficientNet-B0	DeepLabV3	4.04M	3.30M	7.34M	13.97	10.23 ms/img
MobileNetV2	DeepLabV3	2.25M	10.42M	12.68M	51.47	7.33 ms/img

Table 4.2: summary of all the Econder-Decoder combinations tested along with their parameters count (in millions) broken down into encoder, decoder, total model size, GFLOPs and Inference time for the CropAndWeed dataset.

We investigate combinations of lightweight but effective encoder–decoder architectures for semantic segmentation in agriculture. Since efficiency is a factor for real-world deployment, especially in resource-constrained environments, we place special emphasis on evaluating models that balance segmentation accuracy with computational cost. To this end, we tested various architectural combinations that not only enhance performance but also remain within the realm of being usable on mobile devices, ensuring practical applicability in field conditions. As encoders, we consider two widely adopted backbone networks: MobileNetV2 and EfficientNet-B0, both designed to balance accuracy with computational efficiency, making them well suited for edge-device deployment. To assess how encoder choice influences segmentation performance, we systematically pair them with three decoder architectures: UNet, UNet++, and DeepLabV3. UNet serves as our baseline decoder thanks to its proven effectiveness and simplicity in segmentation tasks. UNet++ extends this design with nested and dense skip connections to enhance feature fusion and boundary refinement. Finally, DeepLabV3 introduces atrous spatial pyramid pooling to capture multi-scale context, making it particularly suitable for heterogeneous agricultural scenes (See Sections 2.4 and 2.5). By testing all combinations of the two encoders and three decoders, we establish a comparative framework that highlights the trade-offs between segmentation accuracy, parameter efficiency, and suitability for real-world agricultural applications. Tables 4.1 and 4.2 summarize the parameter counts for each configuration across the WE3DS and CropAndWeed datasets, allowing fair and transparent comparison. All encoder backbones used in this work (MobileNetV2 and EfficientNet-B0) are initialized with weights pretrained on the ImageNet dataset.

4.4 Hyperparameters selection

Hyperparameters play an important role in the training of deep learning models, as they directly influence how the network learns from data and how effectively it converges toward an optimal solution. Unlike model parameters, which are learned automatically during training, hyperparameters must be set before the training process begins and remain fixed throughout a given run. They govern aspects such as the learning rate, batch size, choice of optimizer, and loss functions, all of which can substantially affect model performance. Choosing appropriate hyperparameters is therefore critical to achieving a balance between accuracy, stability, and computational efficiency [59]. Poorly selected values can lead to issues such as slow convergence, overfitting, or underfitting, while well-chosen ones can significantly improve both training speed and final segmentation quality.

Several strategies exist to determine optimal hyperparameters. Grid search explores all possible combinations within a predefined space, ensuring exhaustive coverage but at a high computational cost [21]. Random search samples configurations stochastically and has been shown to be more efficient in practice, often yielding competitive results with fewer trials [3]. More advanced approaches, such as Bayesian optimization, employ probabilistic models to balance exploration and exploitation, efficiently guiding the search toward promising regions of the hyperparameter space [73]. While these methods are powerful, they typically demand considerable computational resources.

In this work, we adopt a manual hyperparameter selection strategy. This choice is motivated by the large number of encoder–decoder combinations tested. Applying automated search techniques to each architecture would have been computationally prohibitive. Instead, we rely on best practices estab-

lished in the literature and insights from preliminary experiments to guide our decisions. This approach provides a pragmatic compromise, ensuring consistency and reproducibility across experiments while allowing a strategic allocation of computational resources.

4.4.1 Loss functions

The loss function is a central component of deep learning models, as it quantitatively measures the discrepancy between the model's predictions and the ground-truth labels. By guiding the optimization process, it directly influences how the network updates its parameters during training. In semantic segmentation tasks, where predictions are made at the pixel level, the choice of loss function plays a role in addressing common challenges such as class imbalance, boundary precision, and small object detection. Different loss formulations emphasize different aspects of the prediction quality. For example, some prioritize global accuracy, while others explicitly compensate for skewed class distributions or focus on improving overlap between predicted and true regions. To investigate these trade-offs in the context of vegetation cover estimation, we evaluate and compare four widely adopted loss functions in the context of multiclass problems and unbalanced data:

- Cross Entropy Loss.
- Weighted Cross Entropy Loss.
- Focal Loss.
- Dice Loss.

Each of these functions has distinct properties that make it suitable for handling specific segmentation challenges. In the following subsections, we describe their theoretical formulation and practical relevance to our task.

Cross Entropy Loss

Cross Entropy Loss (CE) [47] is the most commonly used objective function for classification and segmentation problems. It measures the dissimilarity between the predicted probability distribution and the true class labels by penalizing incorrect predictions more strongly as their confidence increases. For a multiclass segmentation problem with C classes, CE is defined as:

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\,\log\left(p_{i,c}\right)$$

where N denotes the number of pixels in the image, $y_{i,c}$ is a binary indicator (1 if pixel i belongs to class c, otherwise 0), and $p_{i,c}$ is the predicted probability that pixel i belongs to class c. The main advantage of Cross Entropy is its simplicity and effectiveness in encouraging correct class predictions. However, in agricultural segmentation tasks such as crop–weed discrimination, the dataset is often highly imbalanced, with soil pixels dominating the image. In such cases, CE may bias the model toward majority classes, leading to suboptimal performance on minority categories. Despite this limitation, CE serves as a baseline loss function and provides a reliable benchmark against which the performance of more specialized loss functions can be assessed.

Weighted Cross Entropy Loss

One limitation of the standard Cross Entropy Loss is its tendency to favor majority classes, especially in datasets characterized by severe class imbalance. In agricultural segmentation tasks, soil pixels often dominate the image, while minority classes such as weeds occupy relatively few pixels. This imbalance can lead to biased predictions where the model achieves high overall accuracy but fails to capture small or underrepresented categories. To mitigate this issue, Weighted Cross Entropy (WCE) [26] introduces class-specific weights into the loss formulation, assigning higher importance to minority classes and lower importance to majority ones. The loss is defined as:

$$\mathcal{L}_{WCE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} w_c\, y_{i,c}\, \log\left(p_{i,c}\right)$$

where w_c denotes the weight assigned to class c. By setting w_c inversely proportional to class frequency, the loss function penalizes errors on underrepresented classes more strongly, encouraging the model to learn balanced representations across categories. This weighting mechanism makes WCE particularly relevant for vegetation cover estimation, where the accurate segmentation of minority classes (e.g., specific weed species) is essential despite their low pixel prevalence. However, determining optimal weights requires careful tuning: excessively high weights for rare classes may destabilize training, while insufficient weighting may fail to correct the imbalance. In this work, WCE is adopted as one of the tested alternatives to standard Cross Entropy, allowing us to explicitly assess the impact of class balancing on segmentation performance.

Specifically, the weights for the Weighted Cross Entropy were derived directly from the training set annotations. First, all segmentation masks were scanned to count the number of pixels belonging to each class. This produced a frequency distribution that revealed the strong imbalance typical of agricultural images (with soil pixels largely dominating). To counteract this, class weights were computed using an inverse-frequency strategy with logarithmic scaling, which avoids extreme values for rare classes while still increasing their relative importance. These weights were then normalized and passed as a parameter to the PyTorch CrossEntropyLoss function, ensuring that minority classes contributed more strongly to the optimization process. Formally the weights computation can be summarized by these formulas: let N be the number of images in the dataset, C the number of classes, and $I = \{1..N\}$ the set of image indices. For each image $i \in I$ the segmentation mask is $M_i : \Omega \to \{0, ..., C-1\}$ where Ω is the set of pixel coordinates (resized to 256×256) we have:

1. The total pixel count for each class c is (where $\mathbf{1}[\cdot]$ is the indicator function.)

$$n_c = \sum_{i=1}^{N}\sum_{p\in\Omega} \mathbf{1}\big[M_i(p) = c\big]$$

2. The total number of pixels across the dataset is

$$N_{\text{tot}} = \sum_{c=0}^{C-1} n_c$$

3. The inverse frequency for each class is (where ε is a small constant (10^{-6}) to avoid division by zero.

$$f_c^{\text{inv}} = \frac{N_{\text{tot}}}{n_c + \varepsilon}$$

4. The weight for each class used in the weighted cross-entropy is (all weights are collected into a vector w)

$$w_c = \log\left(1 + \frac{N_{\text{tot}}}{n_c + \varepsilon}\right), \qquad c = 0, \ldots, C-1$$

Focal Loss

While Weighted Cross Entropy reduces the bias toward majority classes by applying static weights, it does not specifically address the challenge posed by hard-to-classify examples, which are often under-represented in the dataset. In agricultural segmentation tasks, these cases typically correspond to small weeds or occluded plants, where prediction confidence is low. Standard Cross Entropy (and its weighted version) may allow the model to focus excessively on well-classified majority pixels (e.g., soil), neglecting minority or ambiguous regions. Focal Loss [42] was designed to tackle this issue by dynamically down-weighting the contribution of easy examples and focusing training on hard examples. Its formulation extends the Cross Entropy Loss with a modulating factor:

$$\mathcal{L}_{FL} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} \alpha_c (1 - p_{i,c})^{\gamma}\, y_{i,c} \log\left(p_{i,c}\right)$$

where α_c is a weighting factor to balance class importance, $p_{i,c}$ is the predicted probability for class c at pixel i and $\gamma \geq 0$ is the focusing parameter which reduces the loss contribution from well-classified pixels (high $p_{i,c}$) and amplifies it for poorly classified pixels (low $p_{i,c}$). When $\gamma = 0$, the Focal Loss reduces to Weighted Cross Entropy Loss. For $\gamma > 0$, the effect of the modulating factors increases, enabling the model to focus more on hard samples. This property makes Focal Loss particularly suitable for imbalanced segmentation problems such as vegetation cover estimation. It ensures that the network not only balances class frequency but also pays closer attention to difficult pixels, thereby improving boundary precision and minority class detection. However, selecting appropriate values for α_c and γ is critical: excessively high focusing may destabilize training, while too low values may render the function equivalent to standard Cross Entropy.

For the implementation of Focal Loss, a custom PyTorch class was defined to extend the standard cross-entropy formulation. The logits predicted by the network were first converted into log-probabilities, from which the true-class probabilities were extracted for each pixel. A modulating factor of the form $(1 - p_t)^{\gamma}$ was then applied, where p_t is the predicted probability for the ground-truth class and γ is the focusing parameter. In this work, $\gamma = 2.0$, which effectively reduces the contribution of well-classified pixels and shifts the optimization focus toward harder, misclassified examples. Optionally, class-specific weights (α) could be incorporated in the same framework, allowing the loss to combine the benefits of Focal Loss with those of Weighted Cross Entropy. However, unless otherwise specified, training was performed without additional weighting. The implementation also supported the exclusion of undefined labels through an *ignore_index* mechanism, ensuring that ambiguous pixels in the dataset did not affect gradient updates.

Dice Loss

Unlike Cross Entropy and its variants, which operate on pixel-wise classification probabilities, Dice Loss [69] is based on a region-overlap measure between the predicted segmentation and the ground truth. It derives from the Dice Similarity Coefficient (DSC), also known as the F1-score in binary classification, which evaluates how well two sets of pixels overlap. For two sets A (predicted positives) and B (ground truth positives), the Dice coefficient is defined as:

$$DSC = \frac{2\,|A \cap B|}{|A| + |B|}$$

In practice, for soft predictions in semantic segmentation, the Dice Loss is expressed as:

$$\mathcal{L}_{Dice} = 1 - \frac{2\sum_{i=1}^{N} p_i\, y_i + \epsilon}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} y_i + \epsilon}$$

where p_i is the predicted probability for pixel i, y_i is the corresponding ground-truth label, N is the total number of pixels, and ϵ is a small constant added for numerical stability. Dice Loss directly optimizes for the overlap between predicted and true regions, making it especially effective in highly imbalanced scenarios, where minority classes (e.g., small weeds) contribute little to the overall loss in Cross Entropy. By emphasizing the relative overlap rather than absolute pixel counts, Dice Loss can improve segmentation quality on underrepresented classes and sharpen boundary predictions. However, Dice Loss can be less stable during training, especially at the beginning when predictions are poor, since small denominators may lead to large gradient fluctuations. For this reason, it is often combined with Cross Entropy or other pixel-wise losses to balance stability and overlap optimization. In this work, we evaluate Dice Loss individually to assess its effectiveness in handling class imbalance in agricultural segmentation tasks. Its ability to prioritize overlap makes it a strong candidate for improving the segmentation of minority vegetation classes.

The Dice Loss was implemented by extending the standard Sørensen–Dice coefficient into a differentiable loss function. Model outputs were first normalized with a softmax function to obtain per-class probability maps. Ground-truth masks were then converted into one-hot encoded tensors to align with the multi-class prediction format. For each class, the intersection between predictions and ground truth was computed, and the Dice score was calculated as the ratio between twice the intersection and the sum of predicted and true areas, with a small ϵ added to prevent division by zero. The final Dice loss was defined as $1 - Dice\_Score$, averaged across all classes. This design ensures that the loss directly optimizes for region overlap, making it particularly effective in handling class imbalance and improving segmentation of minority regions.

4.4.2 Optimizer

The optimization of network parameters during training was carried out using the Adam optimizer (Adaptive Moment Estimation) [34] a method that combines the benefits of both momentum and adaptive learning rates. Adam extends stochastic gradient descent by maintaining exponentially decaying averages of past gradients (first moment) and squared gradients (second moment), which allows it to adaptively scale the learning rate for each parameter. This property makes it particularly effective in

handling sparse gradients and accelerating convergence compared to classical optimizers such as standard SGD [62]. In this work, Adam was chosen because of its robustness and efficiency across different network architectures, especially when dealing with heterogeneous datasets characterized by strong class imbalance and high variability. The learning rate was set to $\eta = 10^{-3}$, which offered a good trade-off between convergence speed and stability in preliminary experiments. Default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ were used, following common practice in the literature. This choice of optimizer ensured stable training dynamics across all tested encoder–decoder combinations and loss functions, reducing the need for extensive learning rate scheduling while still providing reliable convergence.

4.4.3 Batch size

The choice of the batch size represents another important step in the training of deep neural networks since it influences both convergence behavior and generalization performance. While training with the entire dataset in a single step (full-batch training) would in principle be feasible given the dataset size and image resolution, this approach is generally avoided as it limits the stochasticity of the optimization process. Instead, mini-batch training offers a practical trade-off, combining computational efficiency with the regularizing effect of gradient noise. Prior studies have shown that larger batch sizes tend to converge more quickly but often lead to sharper minima in the loss landscape, which can result in poorer generalization to unseen data [33] [48]. Conversely, smaller batch sizes introduce higher gradient variance, acting as an implicit form of regularization that encourages the optimizer to explore flatter and more robust minima, typically associated with better generalization properties.
In this work, we systematically explored multiple batch sizes — specifically 8, 16, 32, and 64 — across all model configurations. This experimental setup allowed us to directly evaluate the impact of batch size on both convergence stability and segmentation performance. The comparative results and identification of the most effective batch size are presented and discussed in Chapter 5.

4.5 Evaluation method

To comprehensively assess the segmentation performance of the tested models, we employed both quantitative metrics and qualitative analysis. The evaluation was conducted on the held-out test set, ensuring that no training or validation images were used during performance assessment. Following standard practice in semantic segmentation, we computed the following metrics at the pixel level: Intersection over Union, precision, recall and F1-score. Metrics were computed per class and then averaged to highlight both global performance and the ability of the models to capture minority vegetation categories. In addition to numerical evaluation, segmentation masks produced by the models were visually compared to the corresponding ground-truth annotations. This analysis focused on boundary sharpness, recognition of small and sparse weed species, and robustness under challenging conditions such as occlusion or overlapping crops. By combining quantitative metrics with visual inspection, we ensured a balanced and reliable evaluation of each encoder–decoder configuration across both datasets.

4.5.1 Metrics

For the quantitative evaluation of segmentation performance, we rely on a set of well-established classification metrics applied to the held-out test set. While these measures were initially designed for binary classification tasks, in this work they are extended to multiclass semantic segmentation by computing the score separately for each class and then averaging the results across all classes (macro-averaging). We define the following notation useful for the next formulas:

- TP_c: denotes the number of pixels correctly predicted as belonging to class c (true positives).
- FP_c: denotes the number of pixels incorrectly predicted as belonging to class c (false positives).
- FN_c: denotes the number of pixels belonging to class c but missed by the model (false negatives).

Intersection over Union (Iou)

The Intersection over Union is one of the most widely used metrics for evaluating semantic segmentation performance. For a given class c IoU is defined as:

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c}$$

The metric quantifies the overlap between the predicted segmentation mask and the ground truth mask. A higher IoU indicates a stronger agreement between prediction and ground truth. It is particularly important in semantic segmentation because it penalizes both types of errors—over-segmentation and under-segmentation—making it more robust than simple accuracy, which may be biased toward majority classes (e.g., soil). In our experiments, IoU was computed per class and then aggregated across classes using the macro-average. The implementation begins by obtaining model predictions through an *argmax* operation over the class dimension of the output tensor, which assigns each pixel to the most likely class. Both the predicted masks and the ground-truth labels are then flattened into one-dimensional vectors to enable direct pixel-wise comparison. For each class, the intersection is calculated as the number of pixels where prediction and ground truth coincide, while the union corresponds to the number of pixels assigned to that class in either prediction or ground truth. The IoU score is obtained as the ratio between intersection and union. In cases where a class is absent from both prediction and ground truth, the union is empty; in such cases, the IoU is set to NaN and later ignored during averaging. During validation, IoU values are computed batch by batch, accumulated across the dataset, and finally averaged to obtain both the per-class IoUs and the global mean IoU. This ensures that the reported results reflect performance not only on dominant categories such as soil but also on minority classes such as less frequent weed species.

Precision

Precision is a metric that quantifies the proportion of correctly predicted pixels for a given class among all pixels that the model has labeled as belonging to that class. Formally, for a class c, precision is defined as:

$$Precision_c = \frac{TP_c}{TP_c + FP_c}$$

High precision indicates that the model makes relatively few false positive errors, meaning that when it predicts the presence of a class, it is usually correct. This property is particularly important in crop–weed segmentation, as excessive false positives for weeds could lead to unnecessary interventions and potential damage to crop plants. In our implementation, precision was computed using the *torchmetrics.Precision* class, configured for multiclass segmentation with per-class averaging. The global recall score was obtained by averaging across all classes (macro-average), ensuring that minority classes received the same weight as dominant ones. This approach ensures that the metric reflects the performance on both frequent categories, such as soil, and rare classes, such as minority weed species, thereby providing a balanced and reliable estimate of the model's ability to avoid false positive errors.

Recall

Recall measures the proportion of correctly identified pixels of a given class with respect to all the true pixels of that class present in the ground truth. For a class c, recall is formally defined as:

$$Recall_c = \frac{TP_c}{TP_c + FN_c}$$

High recall indicates that the model successfully captures the majority of the relevant pixels for a class, even if it sometimes produces more false positives. In the context of crop–weed segmentation, recall is important for ensuring that weeds are not overlooked, since missing them could result in reduced yields or uncontrolled weed proliferation. Recall was implemented using the *torchmetrics.Recall* class, configured for multiclass segmentation with per-class computation. The global recall score was obtained by averaging across all classes (macro-average), ensuring that minority classes received the same weight as dominant ones. This procedure allowed the metric to fairly reflect the model's capacity to detect both common categories, such as soil, and underrepresented vegetation classes, particularly weeds that occupy fewer pixels in the images.

F1-score

The F1-score is the harmonic mean of precision and recall, combining both metrics into a single measure that balances false positives and false negatives. For a given class c, it is defined as:

$$Recall_c = \frac{2 \cdot Precision_c \cdot Recall_c}{Precision_c + Recall_c}$$

This formulation ensures that the F1-score is high only when both precision and recall are high, making it particularly informative in scenarios with class imbalance. In the context of crop–weed segmentation, the F1-score provides a balanced evaluation: it penalizes models that achieve high precision at the cost of low recall (or vice versa), thereby highlighting architectures that consistently detect weeds and crops while minimizing both false positives and false negatives. The F1-score was implemented using the *torchmetrics.F1Score* class, configured for multiclass tasks with per-class evaluation. The global score was calculated as the macro-average across all classes. This ensured a fair evaluation of both dominant classes, such as soil, and minority categories, such as specific weed species, providing a robust overall indicator of segmentation quality.

5

Results

This chapter presents and discusses the experimental findings obtained from the semantic segmentation framework described in Chapter 4. Building upon the datasets, model architectures, and training procedures detailed in Chapter 3, we now evaluate how the proposed combinations of encoders, decoders, and loss functions perform in practice.

We begin by outlining the experimental setup, including hardware specifications, training protocols, and implementation details that ensure reproducibility. Next, we analyze the impact of training choices through a batch size analysis and a systematic loss function comparison, highlighting how these factors influence convergence, stability, and segmentation accuracy. The chapter then reports quantitative results for both benchmark datasets. Separate subsections provide overall metrics and class-specific performance for the WE3DS and CropAndWeed datasets, enabling a direct comparison of model behavior across diverse agricultural scenarios. Finally, we complement these numerical evaluations with a qualitative analysis, showcasing visual examples that illustrate each model's strengths and limitations in realistic field conditions. Together, these sections provide a comprehensive assessment of the proposed approach, linking the methodological decisions of the previous chapter to concrete evidence of their effectiveness in precision agriculture.

5.1 Experimental setup

The dataset was split into training, validation, and test sets according to the procedure described in Section 4.2. Six different architecture–backbone combinations were evaluated. All models were trained under identical conditions—using the same loss function, optimizer, learning-rate schedule, and batch size outlined in Chapter 4—to ensure a fair comparison. Model performance was assessed with four widely used semantic-segmentation metrics: Intersection over Union, precision, recall, and the F1 score (also known as the Dice Similarity Coefficient). Metrics were computed for each class and then macro-averaged, providing both class-level and overall perspectives, which is particularly important for evaluating performance on rare or difficult classes. All experiments were conducted on a Linux server equipped with an NVIDIA RTX A5000 GPU (24 GB VRAM) using PyTorch 2.4.1 and CUDA 11.8.

5.2 Batch size analysis

To determine an effective mini-batch size for training, we conducted a dedicated set of experiments on the WE3DS dataset. As described in Section 4.4.3, batch size influences both the stability of gradient updates and the computational efficiency of the training process. Testing a range of configurations exclusively on WE3DS allowed us to balance these factors while keeping the hyperparameter search manageable. The goal of this analysis was to identify the best batch size that ensured stable convergence without sacrificing segmentation accuracy. After comparing the resulting learning curves and validation metrics across the different configurations, we selected the best-performing setting as our default. Once this optimal value was established, we adopted the same batch size strategy for the CropAndWeed dataset without repeating the full set of experiments. This choice was motivated by two considerations:

- The selected batch size had already demonstrated robust training behavior on WE3DS.
- Replicating the entire search on the larger and more complex CropAndWeed dataset would have imposed a substantial computational overhead with little expected benefit.

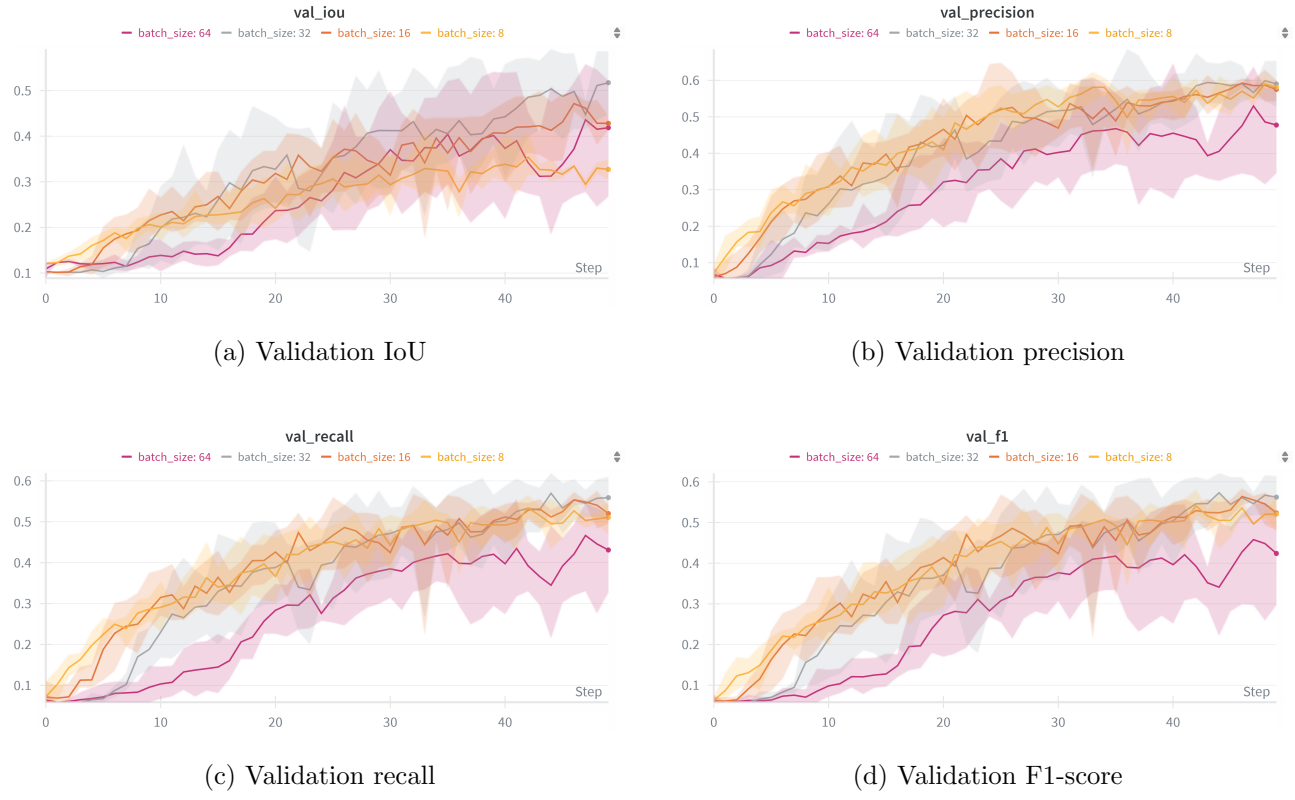
Figure 5.1a shows the evolution of the validation IoU for the different batch sizes tested on the WE3DS dataset. Among the four configurations—8, 16, 32, and 64—the best performance was achieved with a batch size of 32, reaching the highest and most stable IoU across training steps. The smallest batch size (8) consistently underperformed, indicating insufficient gradient stability. A similar trend emerged for the other metrics (precision depicted in Figure 5.1b, recall depicted in Figure 5.1c, and F1 score depicted in Figure 5.1d): batch size 32 again yielded the highest scores, while the remaining configurations performed slightly lower but followed a comparable pattern. Notably, for these three metrics the worst results were obtained with batch size 64, suggesting that overly large batches may limit generalization in this task. Based on these findings, a batch size of 32 was selected as the optimal configuration and used for all subsequent experiments.

5.3 Loss function analysis

To investigate the impact of different loss functions on segmentation performance, we conducted a systematic comparison of Cross-Entropy, Weighted Cross-Entropy, Focal Loss, and Dice Loss, as described in Section 4.4.1. All experiments were performed under identical training conditions, using the same encoder–decoder architectures, learning rate (set to 10e-3), optimizer (Adam), and batch size (set to 32). Since the WE3DS and CropAndWeed datasets differ significantly in terms of class distribution, number of categories, and level of imbalance, we present the loss function results separately for each dataset. This allows a clearer interpretation of the impact of each loss function under different levels of complexity and class skew, and highlights how the findings generalize across datasets.

5.3.1 WE3DS analysis

On the WE3DS dataset, all four loss functions converged to stable solutions within 50 epochs. Figures 5.2a, 5.2b, 5.2c and 5.2d reports the macro-averaged IoU, Precision, Recall, and F1-score across the 19 classes of the WE3DS test set, while Table 5.1 the highest validation metrics achieved during training.



(a) Validation IoU

(b) Validation precision

(c) Validation recall

(d) Validation F1-score

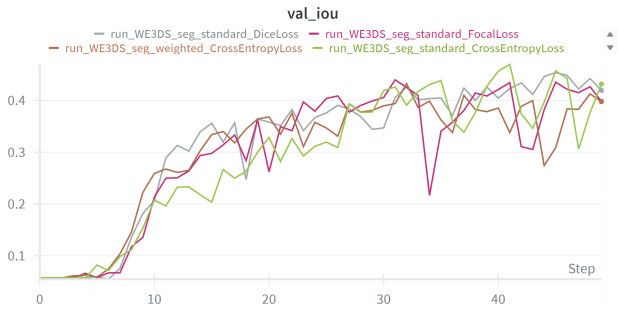
Figure 5.1: Validation metrics (IoU, precision, recall, F1-score) for different batch sizes on the WE3DS dataset.

Loss Function	IoU	Precision	Recall	F1-score
CE	**0.4694**	0.6073	0.5660	0.5690
WCE	0.4323	0.5133	**0.6322**	0.5513
Focal	0.4396	0.5816	0.5299	0.5386
Dice	0.4537	**0.6568**	0.6058	**0.6120**

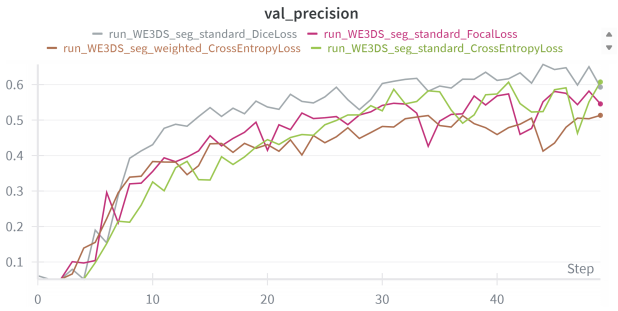
Table 5.1: Highest validation values achieved during training on the WE3DS dataset for the different loss functions.

Overall, the results highlight that no single loss function dominates across all metrics. However, because the F1-score is itself the harmonic mean of Precision and Recall, it can be informative to look at a combined benchmark of IoU and F1 together, or even their simple average, to summarize both overlap and classification balance. Under this joint view, Dice stands out: although Dice and WCE achieve very similar IoU values (difference of about 0.02–0.03) and Recall differs by only about 0.03, Dice's Precision is roughly 0.14 higher. Consequently, whether we average IoU and F1 directly or take a simple row-wise mean of all four metrics, Dice still yields the highest overall score. Cross-Entropy achieves the highest mean IoU, providing a strong and stable baseline. Weighted Cross-Entropy significantly boosts recall, as it achieves 0.6322 (about +0.07 compared to CE) confirming its effectiveness in recovering minority classes, though at the cost of lower precision. Dice Loss reaches the best overall F1-score and the highest precision (0.6568), indicating sharper and more consistent segmentation boundaries. These findings demonstrate that reweighting strategies (WCE) and overlap-based losses (Dice) are valuable when the focus is on minority-class recovery or boundary quality, respectively, while CE remains a

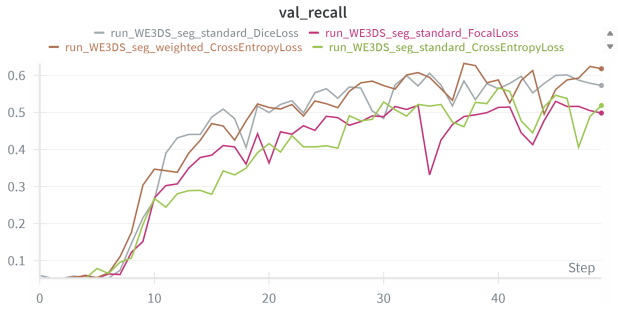
reliable default for balanced performance.



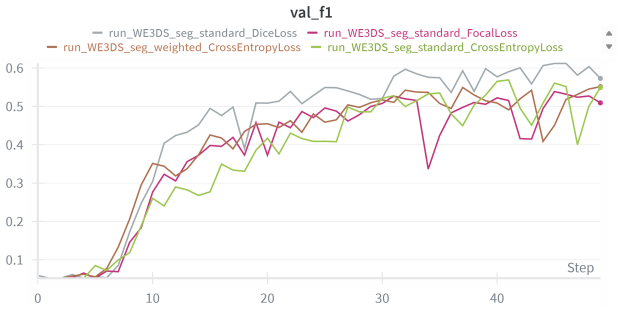
(a) Validation IoU



(b) Validation precision



(c) Validation recall



(d) Validation F1-score

Figure 5.2: Validation metrics (IoU, precision, recall, F1-score) for different loss functions.

The benefits of using weighted losses are most evident on minority classes such as *corn spurry* (class 3 of the dataset), and *common wild oat* (class 8 of the dataset) as depicted in Figures 5.3 and 5.4. For both classes Dice Loss achieves the highest values, reaching peaks above 0.8 and 0.5 respectively, while both Cross-Entropy remains near zero. This illustrates how overlap-based losses such as Dice can substantially improve the detection of rare classes, preventing the network from completely ignoring them. These improvements indicate that reweighting and hard-example mining effectively prevent the model from ignoring rare species.

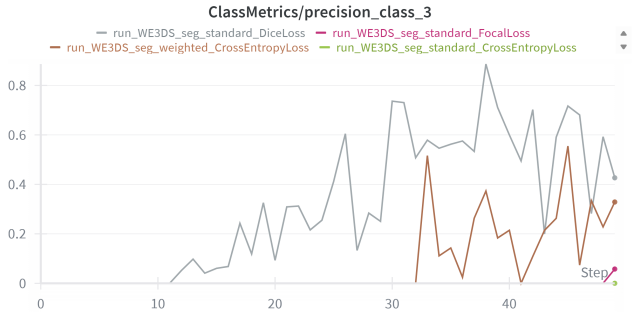


Figure 5.3: Class 3 precision evolution.

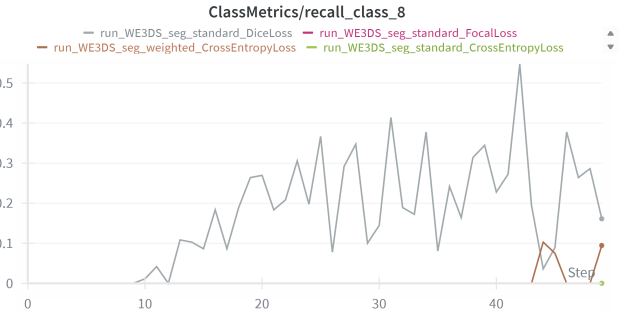


Figure 5.4: Class 8 recall evolution.

5.3.2 CropAndWeed analysis

Compared to WE3DS, the CropAndWeed dataset presents a more challenging scenario, with 101 semantic classes, highly imbalanced class distributions, and substantial visual variability due to diverse field conditions. As a result, the choice of loss function plays an even more important role in ensuring that minority species are not ignored during training. Figures 5.5a, 5.5b, 5.5c and 5.5d report the macro-averaged IoU, Precision, Recall, and F1-score for all loss functions, while Table 5.2 summarizes the best results obtained during training.

Loss Function	IoU	Precision	Recall	F1-score
CE	**0.1436**	0.3017	0.2478	0.2531
WCE	0.1406	0.2782	**0.3305**	**0.2889**
Focal	0.1264	**0.3138**	0.2449	0.2457
Dice	0.1101	0.2029	0.2092	0.1918

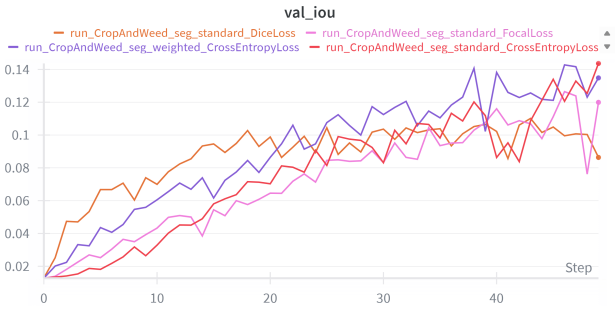
Table 5.2: Comparison of loss functions on the CropAndWeed dataset.

These results confirm that no single loss function achieves superior performance across all metrics. Cross-Entropy remains the strongest baseline, yielding the highest IoU and competitive precision. Weighted Cross-Entropy provides a clear improvement in recall, confirming its effectiveness in enhancing the detection of minority classes—though this comes at the cost of reduced precision. Focal Loss maintains high precision but does not significantly improve recall compared to CE, which suggests that its focus on hard examples does not fully compensate for the extreme class imbalance of this dataset. Dice Loss performs the worst across all metrics, indicating that overlap-based objectives struggle in scenarios with a very large number of highly imbalanced classes. Again, the true benefits of the weighted losses become more evident when analyzing performance on individual classes, where WCE consistently improves the recognition of rare species that would otherwise be neglected. Figures 5.6 and 5.7 shows again this point respectively on classes 95 (*Wild carrot*) and 96 (*Field mustard*) of the dataset.

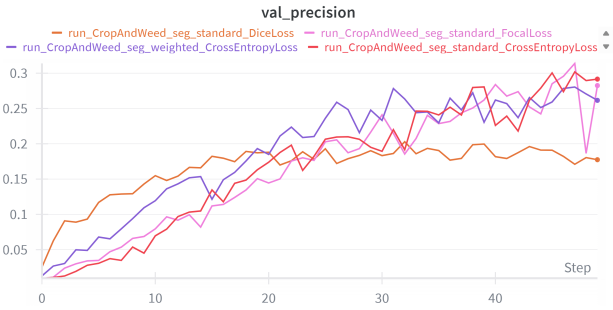
5.4 Best model selection

To identify the most suitable segmentation model, a two–stage procedure was adopted. In the first stage, six encoder–decoder combinations were evaluated on both datasets (as reported in the Section 4.3): MobileNetV2 + UNet, MobileNetV2 + UNet++, MobileNetV2 + DeepLabV3, EfficientNet-B0 + UNet, EfficientNet-B0 + UNet++, and EfficientNet-B0 + DeepLabV3. Each model was trained for 25 epochs with a batch size of 8, employing early stopping with a patience of five epochs to prevent overfitting when no improvement was observed. This preliminary comparison revealed that architectures using UNet or UNet++ consistently outperformed those with DeepLabV3, which were therefore discarded from subsequent experiments.

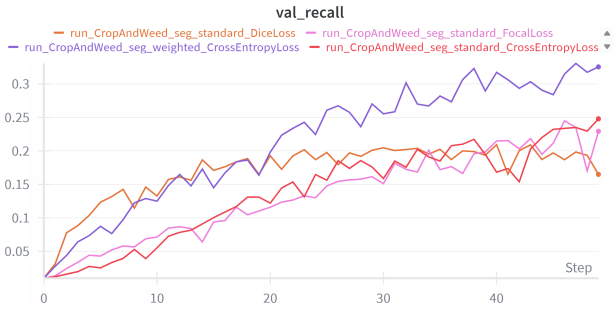
The second stage focused on the four remaining models (MobileNetV2 + UNet, MobileNetV2 + UNet++, EfficientNet-B0 + UNet, and EfficientNet-B0 + UNet++). As detailed in Section 5.2, this phase aimed to explore the influence of batch size using only the WE3DS dataset, which provided a sufficiently rich and balanced benchmark. Here the goal was no longer a quick comparison but a careful examination of batch-size effects.To let each configuration fully converge and to observe the highest attainable per-



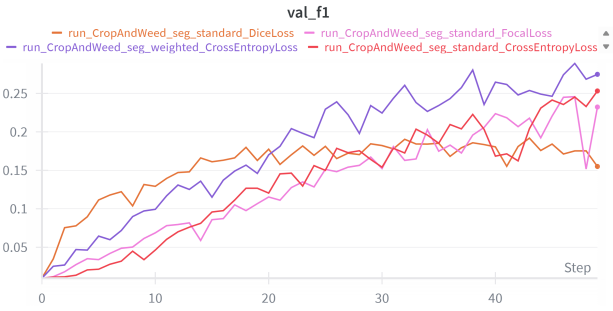
(a) Validation IoU



(b) Validation precision



(c) Validation recall



(d) Validation F1-score

Figure 5.5: Validation metrics (IoU, precision, recall, F1-score) for different loss functions.

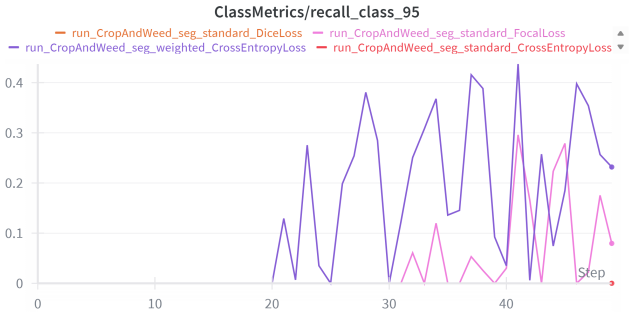


Figure 5.6: Class 95 recall evolution.

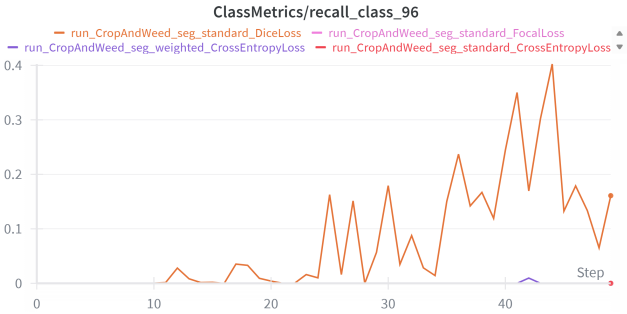


Figure 5.7: Class 96 recall evolution.

formance, training was extended to 50 epochs and early stopping was disabled. The goal of this longer schedule was to see how far the models could improve once the best architecture family was known. The results of this training showed that EfficientNet-B0 + UNet with a batch size of 32 achieved the best overall performance, offering the highest metrics values and stable training dynamics. Consequently, this configuration was selected as the reference model for all subsequent quantitative and qualitative analyses presented in the following sections.

5.5 WE3DS Quantitative results

The quantitative evaluation on the WE3DS dataset highlights distinct performance differences across the tested loss functions (Table 5.3). Cross-Entropy served as a solid baseline, reaching a mean IoU of 0.4694 and an F1-score of 0.5690, which indicates a relatively balanced trade-off between precision and

recall. Weighted Cross-Entropy, despite being explicitly designed to mitigate class imbalance, underperformed compared to CE in terms of IoU (0.3996) and F1 (0.5420). This suggests that in the WE3DS setting, where soil pixels dominate the image space, the weighting strategy may have overcompensated, penalizing the model's ability to generalize across minority classes. Focal Loss also produced suboptimal results, with the lowest recall (0.4886) and a modest F1-score (0.5077). Its emphasis on hard-to-classify pixels may have led the model to focus excessively on ambiguous regions, at the expense of consistent segmentation across the field. On the other hand, Dice Loss achieved the highest precision (0.6474) and F1-score (0.6120), coupled with a recall value of 0.6011, showing that its overlap-based optimization is better suited to dealing with class imbalance and boundary regions. This is consistent with prior literature, where Dice is often preferred in agricultural imaging for handling highly skewed pixel distributions [75] [19].

Loss Function	**IoU**	**Precision**	**Recall**	**F1-score**
CE	**0.4694**	0.6067	0.5566	0.5690
WCE	0.3996	0.5055	**0.6131**	0.5420
Focal	0.4056	0.5413	0.4886	0.5077
Dice	0.4484	**0.6474**	0.6011	**0.6120**

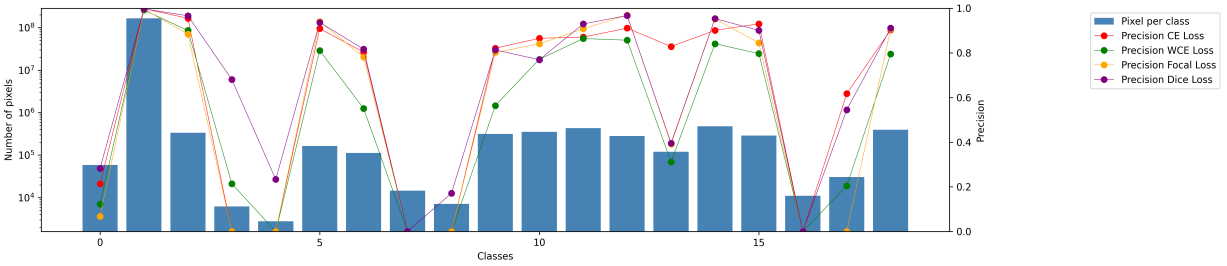
Table 5.3: WE3DS quantitative results.

These results indicate that while CE remains a strong general-purpose choice, Dice Loss provides superior performance for the WE3DS dataset, especially when considering precision–recall balance. WCE and Focal, although theoretically motivated to handle imbalance, proved less effective in this context. These findings emphasize the importance of empirically testing loss functions rather than assuming that imbalance-focused designs will always outperform standard baselines.

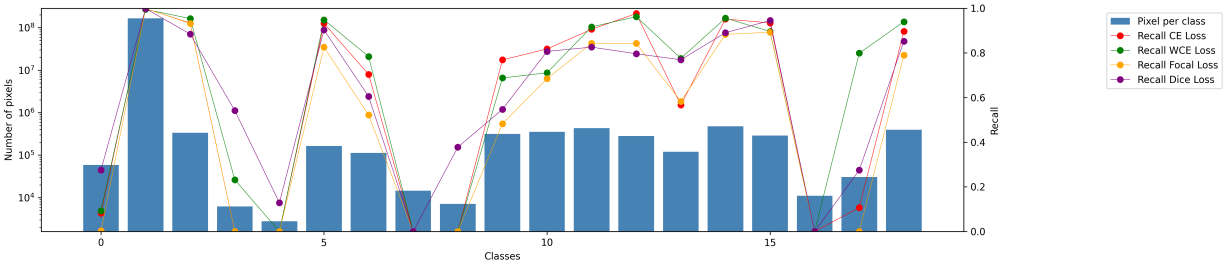
5.5.1 Class-specific results

To further investigate model performance on the WE3DS dataset, we analyzed class-specific results across all 19 semantic categories (soil, crops, and weeds). Figures 5.8a, 5.8b, 5.8c and 5.8d depict the variation of Precision, Recall, F1, and IoU per class, comparing the four evaluated loss functions. The blue bars represent the number of pixels per class, while colored lines trace the performance metrics for each loss. The results confirm that class frequency strongly influences performance. Soil, which accounts for the majority of pixels, achieves consistently high scores across all metrics. Both CE and Dice Loss reached near-optimal IoU and Precision for soil, reflecting the advantage of abundant training samples and the relatively homogeneous visual characteristics of the background. In contrast, minority classes (e.g., rare weeds such as small-flower geranium depicted as the 17th column of the figures) exhibited the lowest performance across all metrics. These classes suffered from limited pixel representation, leading to unstable predictions regardless of loss function. Weighted Cross-Entropy partially improved recall in these classes, but at the expense of precision, highlighting its tendency to increase false positives when rebalancing class weights. The crop classes (e.g., pea, sunflower, sugar beet) generally achieved moderate to high IoU and F1 values. Cross-Entropy produced stable results across crops, while Focal Loss occasionally underperformed due to its aggressive reweighting of hard samples, which reduced performance on more easily distinguishable crop pixels. The weed classes were the most challenging,

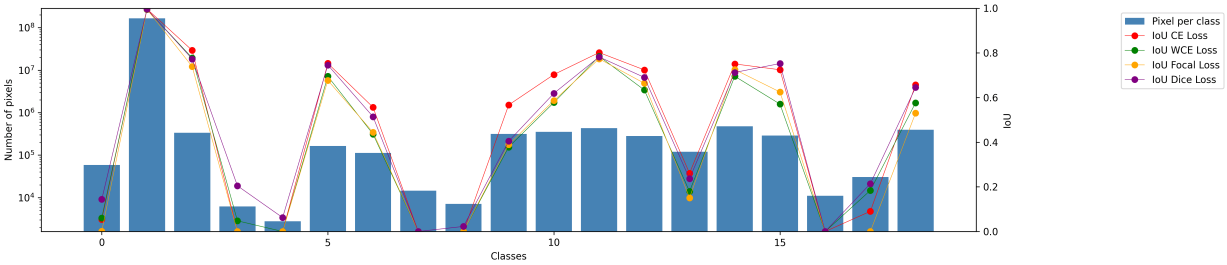
especially when visually similar to crops at early growth stages (e.g., corn spurry vs. corn, rye brome vs. wheat species). Here, Dice Loss achieved the most balanced trade-off between precision and recall, while WCE improved recall but reduced precision, and Focal Loss failed to provide clear advantages. This indicates that while imbalance-focused losses can help avoid missing weed pixels, they risk overestimating weed cover by misclassifying background or crop pixels.



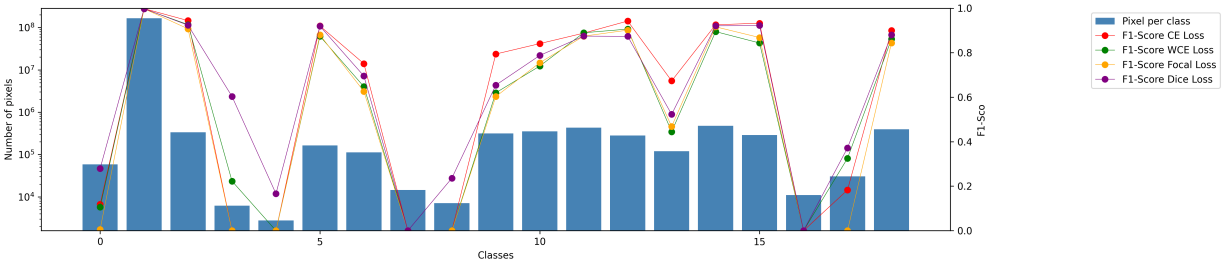
(a) Class-wise Precision.



(b) Class-wise Recall.



(c) Class-wise IoU.



(d) Class-wise F1-Score.

Figure 5.8: Class-wise metrics across all WE3DS classes for the four tested loss functions.

5.6 CropAndWeed Quantitative results

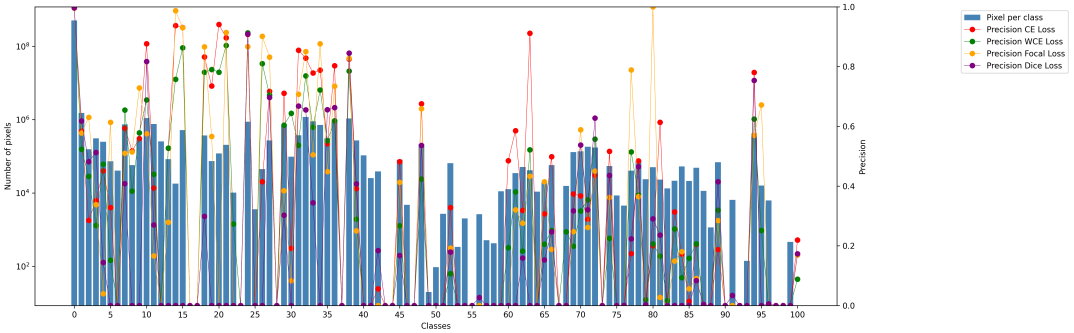
The CropAndWeed dataset presents a substantially more complex challenge compared to WE3DS, due to its higher number of semantic classes, class imbalance, and greater variability in field conditions. This complexity is reflected in the lower overall performance metrics across all loss functions, with mean IoU values not exceeding 0.14 and F1-scores remaining below 0.29 (Table 5.4). Cross-Entropy achieved the highest IoU (0.1329) and a competitive F1-score of 0.2452, confirming its robustness as a baseline even in highly imbalanced and heterogeneous datasets. Weighted Cross-Entropy yielded the best recall (0.3229) and the highest F1-score (0.2803), showing that reweighting helps recover minority classes more effectively. However, this came at the cost of lower precision (0.2722), indicating that while more weed and crop pixels were detected, the model also introduced additional false positives. Focal Loss performed poorly, with IoU and F1 values only slightly above Dice Loss. Despite maintaining precision levels comparable to WCE, its recall dropped sharply (0.2029), suggesting that the emphasis on difficult samples was not sufficient to counteract the dataset's imbalance. Dice Loss delivered the weakest results overall, with the lowest precision (0.1710) and F1-score (0.1752). This indicates that overlap-based optimization, while effective on WE3DS, struggles in large-scale multi-class segmentation where many classes are underrepresented.

Loss Function	**IoU**	**Precision**	**Recall**	**F1-score**
CE	0.1329	**0.3017**	0.2348	0.2452
WCE	**0.1406**	0.2722	**0.3229**	**0.2803**
Focal	0.1087	0.2738	0.2029	0.2067
Dice	0.1007	0.1710	0.1983	0.1752

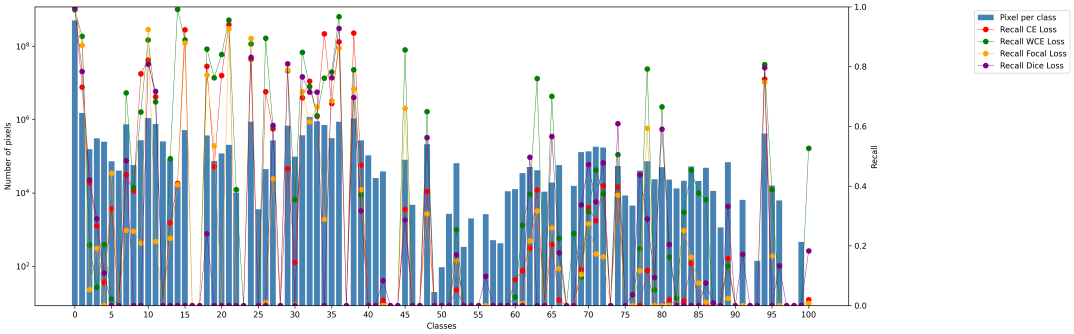
Table 5.4: CropAndWeed quantitative results.

5.6.1 Class-specific results

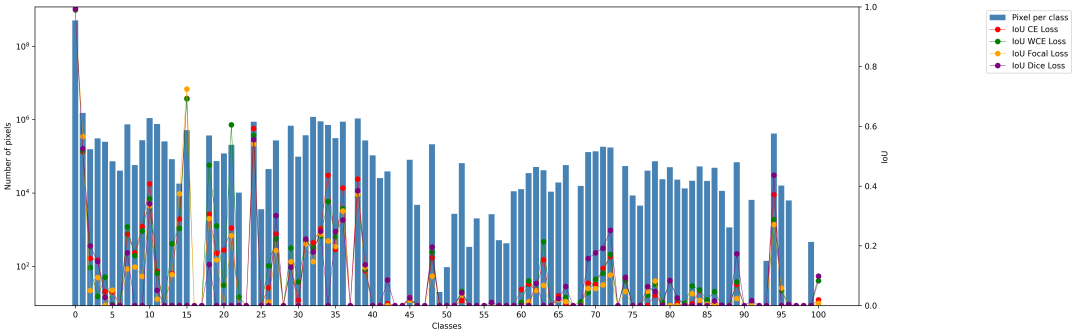
Also in this case, to further investigate model performance on the CropAndWeed dataset, we analyzed class-specific results across all 101 semantic categories. Figures 5.9a, 5.9b, 5.9c, and 5.9d depict the variation of Precision, Recall, IoU, and F1 per class, comparing the four evaluated loss functions. Weighted Cross-Entropy partially improved recall in minority classes, but often at the expense of precision, indicating its tendency to increase false positives when rebalancing class weights. For medium-frequency classes, Dice Loss showed the most consistent improvements across Recall, F1, and IoU, highlighting its strength in optimizing overlap-based metrics and capturing fine object boundaries. Cross-Entropy delivered stable results across the more frequent classes but failed to recover minority-class pixels effectively. Focal Loss produced variable results—sometimes outperforming other losses on very rare classes due to its focus on hard examples, but occasionally underperforming on more easily separable classes because of its aggressive reweighting.



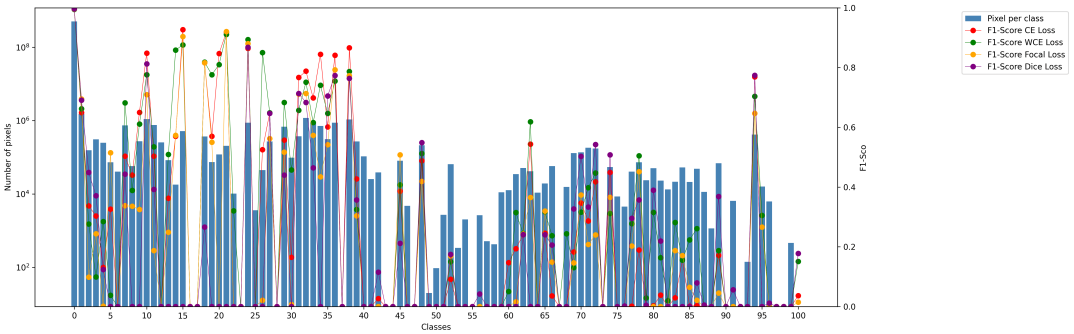
(a) Class-wise Precision.



(b) Class-wise Recall.



(c) Class-wise IoU.



(d) Class-wise F1-Score.

Figure 5.9: Class-wise metrics across all CropAndWeed classes for the four tested loss functions.

5.7 Qualitative analysis

While quantitative metrics provide an objective measure of model performance, qualitative evaluation is equally important for understanding the practical effectiveness of semantic segmentation in precision agriculture. Numerical scores alone may hide systematic errors—for example, consistent misclassification of crop boundaries or subtle weed patches—whereas visual inspection reveals these patterns directly. To assess the visual quality of the predicted segmentation masks, representative test images from both the WE3DS and CropAndWeed datasets were compared against their ground-truth annotations. Figures 5.10 and 5.11 illustrate the outcomes of the different loss functions applied to the same test image.

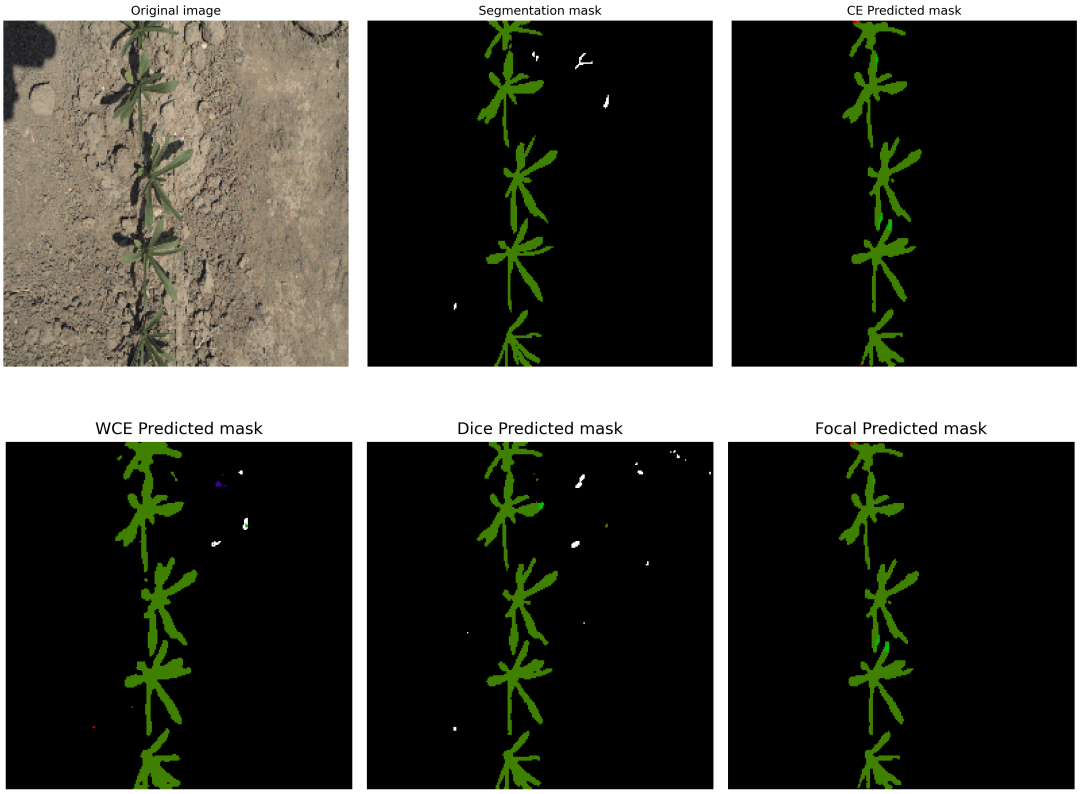


Figure 5.10: Prediction examples on the WE3DS dataset using the model trained with Dice Loss.

5.8 Results on the post-evaluation dataset

To further assess the generalization capability of the proposed segmentation framework, we tested the best performing model on the independent post-evaluation dataset described in Section 3.2. Unlike the WE3DS and CropAndWeed datasets used for training and validation, this collection of high-resolution images was acquired in the University of Udine gardens under uncontrolled field conditions, providing a rigorous test of the model's robustness. Because the original training data did not include lentil plants and contained only early growth stages, the evaluation was restricted to the T1 subset (15 days after emergence). The chosen model was the WE3DS network trained with the Dice loss, which achieved the highest quantitative metrics for buckwheat segmentation in previous experiments. This configuration obtained a Recall of 0.9027, Precision of 0.9367, Intersection-over-Union of 0.7456, and F1-score of 0.9194, making it the most suitable candidate for evaluating buckwheat in the Udine dataset.

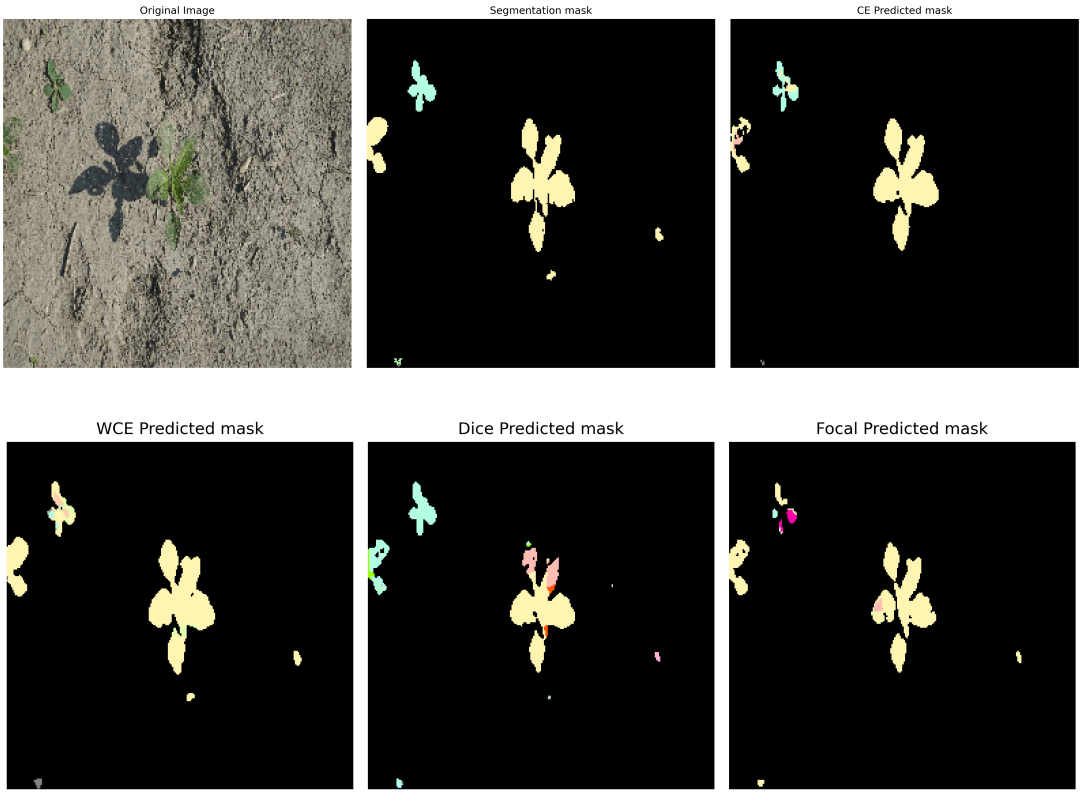


Figure 5.11: Prediction examples on the CropAndWeed dataset using the model trained with WCE.

Given the high native resolution of the images, different prediction strategies were explored to balance computational efficiency and segmentation fidelity:

- Global resizing: each image was uniformly scaled down to 256 × 256 pixels before inference and then upsampled to the original dimension.
- Patch-based inference: the original images were divided into tiles of varying sizes and the predictions recombined into a full-resolution mask.

5.8.1 Quantitative results

The post-evaluation dataset was analyzed to quantify segmentation performance under the different inference strategies described above. For this analysis, all weed species detected by the WE3DS-trained model were merged into a single "Weeds" class, allowing direct comparison with the annotated categories of the Udine dataset. Tables 5.5, 5.6, 5.7 and 5.8 report per-class Precision, Recall, Intersection-over-Union, and F1-score for each prediction strategy.

Class	**1-Patch**	**6-Patches**	**12-Patches**	**35-Patches**
Soil	0.9641	0.9654	0.9762	0.9949
Buckwheat	0.4221	0.5029	0.5210	0.5315
Weeds	0.0384	0.0394	0.0494	0.0572

Table 5.5: Precision metrics values for the different strategies applied to the images.

Class	1-Patch	6-Patches	12-Patches	35-Patches
Soil	0.8655	0.9220	0.9240	0.9376
Buckwheat	0.7628	0.7639	0.7805	0.7794
Weeds	0.0032	0.0042	0.0054	0.0061

Table 5.6: Recall metrics values for the different strategies applied to the images.

Class	1-Patch	6-Patches	12-Patches	35-Patches
Soil	0.8610	0.9140	0.9288	0.9330
Buckwheat	0.4165	0.4959	0.5210	0.5214
Weeds	0.0030	0.0045	0.0061	0.0072

Table 5.7: IoU metrics values for the different strategies applied to the images.

Across all inference strategies, Soil is consistently segmented with high accuracy, reaching Precision and Recall above 0.92 and IoU values up to 0.93. Buckwheat shows moderate but stable performance, with F1-scores ranging from 0.54 to 0.63; the best result is achieved with the 35-patch strategy, suggesting that patch-wise processing helps preserve fine crop details compared to a single global resize. In contrast, the model struggles to identify Weeds, with low metrics across all methods.

Table 5.9 reports the average inference time per image for each prediction strategy. As expected, increasing the number of patches substantially raises the computational cost, since more forward passes through the network are required. While the single-patch approach achieves the fastest inference (317 ms/img), its segmentation quality is generally lower, particularly for small and fine structures. Conversely, the 35-patch strategy delivers the best performance in terms of crop segmentation, but at the cost of an average inference time exceeding 1.7 seconds per image. The 6 and 12 patches strategies provide an intermediate trade-off between efficiency and accuracy, suggesting they may be suitable in scenarios where processing speed is a limiting factor.

5.8.2 Qualitative results

To complement the quantitative evaluation, Figure 5.12 presents representative qualitative results obtained on the post-evaluation dataset. Each example shows the original high-resolution image, the ground-truth annotation, and the predicted segmentation mask for the best-performing WE3DS model. From these visualizations, several observations can be made:

- The model is highly accurate in identifying soil areas, but it also performs well enough to detect crop edges.
- Very small seedlings are correctly detected, as highlighted, for instance, by the orange parts in

Class	1-Patch	6-Patches	12-Patches	35-Patches
Soil	0.9164	0.9471	0.9529	0.9652
Buckwheat	0.5392	0.6070	0.6107	0.6253
Weeds	0.0056	0.0063	0.0071	0.0087

Table 5.8: F1 metrics values for the different strategies applied to the images.

Method	Average Inference time
1-Patch	317 ms/img
6-Patches	567.5 ms/img
12-Patches	775.8 ms/img
35-Patches	1726.3 ms/img

Table 5.9: Average inference time on the images for the proposed methods.

some predictions.

- Crop distinction remains challenging: large regions may be entirely predicted as buckwheat (purple), likely because the other crop type are absent from the WE3DS dataset.
- The presence of orange lines in the middle of predictions is an artifact of patch-based processing without overlap, limiting the model's ability to capture full context.

Several avenues could be explored to further improve segmentation performance. First, fine-tuning on the specific crops present in the target field could help the model better distinguish between crop types that were not included in the original training dataset. Second, adopting a patch-based approach with overlapping windows could reduce artifacts, such as spurious lines in the predictions, by providing the model with more contextual information. Together, these strategies are expected to enhance both crop identification and the accurate detection of small seedlings and crop edges, while minimizing misclassification of weeds and background regions.

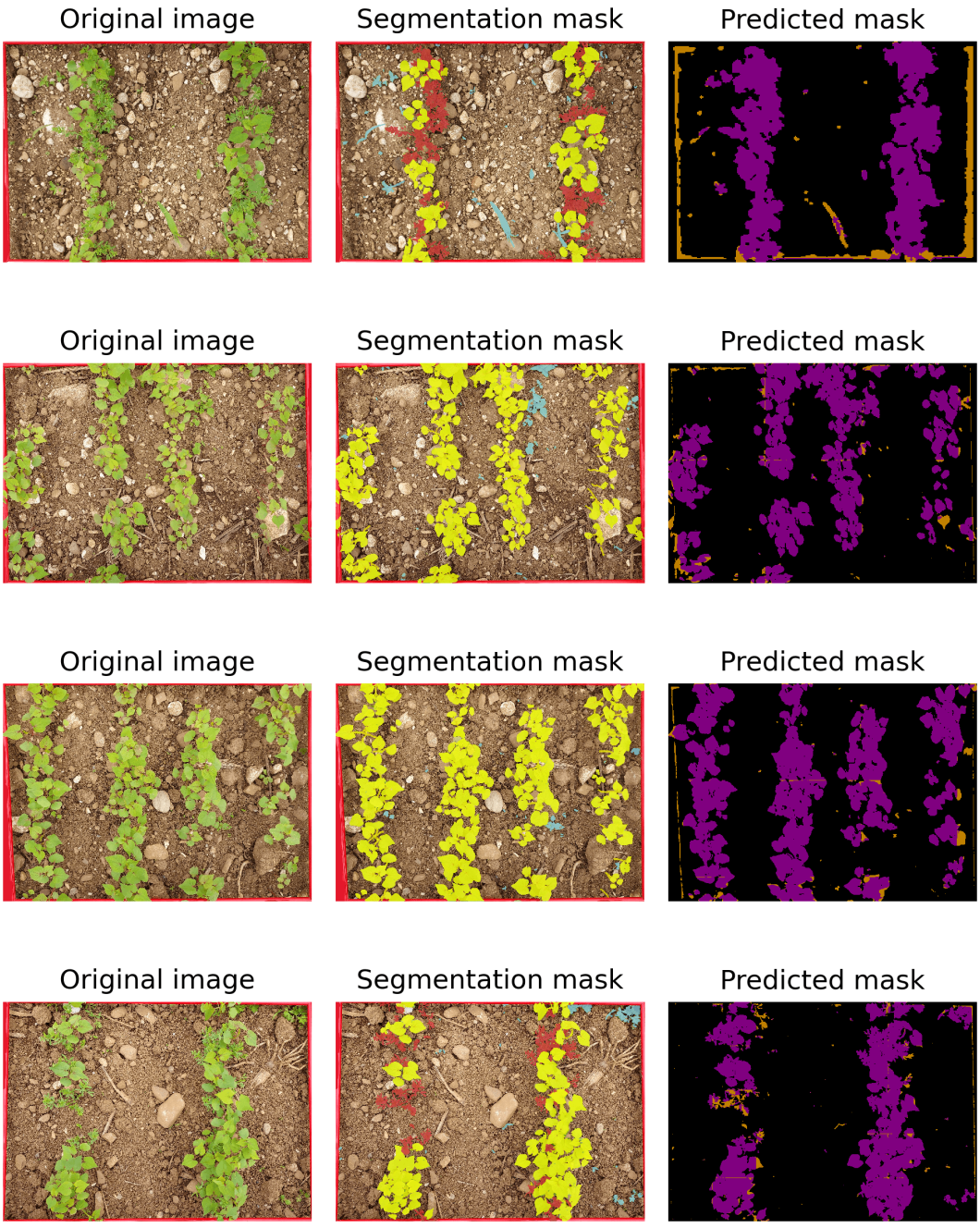


Figure 5.12: Prediction examples on the post-evaluation dataset. The first picture shows the result using the first approach (resizing at 256x256) while the others represent the result of using the patches approach (respectively 6, 12 and 35 patches).

6

Conclusion

In this work, we set out to automate the estimation of vegetation cover in agricultural field settings using deep learning, with the goal of improving accuracy, reproducibility, and scalability compared to traditional manual assessment by also balancing the tradeoff between accuracy and efficiency. Vegetation cover, as a simple yet powerful metric, captures key aspects of crop development, species competition, and field health. However, its manual estimation is inherently limited by human subjectivity, labor demands, and low throughput, particularly in experimental intercropping systems where multiple species coexist, often with significant occlusion and class imbalance. To address these challenges, we framed vegetation cover estimation as a multi-class semantic segmentation task. By predicting dense, per-pixel class labels from RGB images, we aimed to generate maps from which species-specific cover could be directly computed. Our methodological pipeline was designed to ensure reproducibility and scalability by exploring a diverse set of segmentation architectures, systematically comparing models. The results demonstrated that deep learning models, when properly trained, are highly effective at segmenting complex vegetation scenes, even under the constraints of small and imbalanced datasets. Among the architectures tested EfficientNet-B0 + UNet emerged as the best-performing architecture, with an optimal batch size of 32 and Dice loss providing the most stable and accurate segmentation. On the WE3DS dataset, this configuration achieved the highest F1-score (0.612) and IoU (0.469), while on the more challenging CropAndWeed dataset, Cross-Entropy and Weighted Cross-Entropy offered the best robustness despite lower scores (F1 $\leq$ 0.28). When tested on an independent high-resolution dataset from the University of Udine, the selected model maintained strong generalization, reaching 0.92 Precision, 0.90 Recall, and 0.92 F1 for buckwheat segmentation. Patch-based inference further improved fine-detail crop detection, with a 35-patch strategy yielding the best trade-off between accuracy and efficiency. Visual inspection confirmed that the model's predictions were consistent across seen and unseen image regions and its predictions closely matched the annotated ground truth.

Looking forward, several directions could further enhance the utility and robustness of this approach. Expanding the dataset to include additional species, cropping systems, and seasonal snapshots would improve model generalization. Exploring domain-specific or self-supervised pretraining strategies could reduce reliance on manual labels, particularly in large-scale deployments. Integrating complementary sensing modalities, such as multispectral or LiDAR data, would enable richer representations of vegetation structure, especially in dense or layered canopies. Temporal analysis across repeated observations

could help evaluate the consistency of the model's predictions over time and detect meaningful trends in vegetation development, a critical feature for real-world crop monitoring.

This work demonstrated that deep learning–based semantic segmentation is not only a viable but a powerful solution for automating vegetation cover estimation in complex agricultural settings. Through careful model selection, training design, and evaluation strategies, we showed that accurate, class-specific cover metrics can be obtained from simple RGB images, paving the way toward more efficient, objective, and scalable monitoring systems for sustainable agriculture.

Bibliography

[1] Suchet Bargoti and James P Underwood. Image segmentation for fruit detection and yield estimation in apple orchards. *Journal of Field Robotics*, 34(6):1039–1060, 2017.

[2] Hadjer Bechinia, Djamel Benmerzoug, and Nawres Khlifa. Approach based lightweight custom convolutional neural network and fine-tuned mobilenet-v2 for ecg arrhythmia signals classification. *IEEe Access*, 12:40827–40841, 2024.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The journal of machine learning research*, 13(1):281–305, 2012.

[4] Bharath Kumar Bolla, Mohan Kingam, and Sabeesh Ethiraj. Efficient deep learning methods for identification of defective casting products. In *International Conference on Cognition and Recongition*, pages 152–164. Springer, 2021.

[5] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[6] Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

[7] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks*, 106:249–259, 2018.

[8] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020.

[9] Ekin Celikkan, Mohammadmehdi Saberioon, Martin Herold, and Nadja Klein. Semantic segmentation of crops and weeds with probabilistic modeling and uncertainty quantification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 582–592, 2023.

[10] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[11] Fengying Dang, Dong Chen, Yuzhen Lu, and Zhaojian Li. Yoloweeds: A novel benchmark of yolo object detectors for multi-class weed detection in cotton production systems. *Computers and Electronics in Agriculture*, 205:107655, 2023.

[12] Thai-Viet Dang. Smart home management system with face recognition based on arcface model in deep convolutional neural network. *Journal of Robotics and Control (JRC)*, 3(6):754–761, 2022.

[13] Etienne David, Mario Serouart, Daniel Smith, Simon Madec, Kaaviya Velumani, Shouyang Liu, Xu Wang, Francisco Pinto Espinosa, Shahameh Shafiee, Izzat SA Tahir, et al. Global wheat head dataset 2021: An update to improve the benchmarking wheat head localization with more diversity. *CoRR*, 2021.

[14] Michael Day and Arne BR Witt. Weed biological control: challenges and opportunities. *Weeds-Journal of the Asian-Pacific Weed Science Society*, 1(2):34–44, 2019.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[16] Tianmin Deng and Yongjun Wu. Simultaneous vehicle and lane detection via mobilenetv3 in car following scene. *Plos one*, 17(3):e0264551, 2022.

[17] AVISHEK Dutta, JOSEPH M Gitahi, PRASHANT Ghimire, ROBIN Mink, G Peteinatos, J Engels, M Hahn, and R Gerhards. Weed detection in close-range imagery of agricultural fields using neural networks. *Publ. DGPF*, 27:633–645, 2018.

[18] Mads Dyrmann, Anders Krogh Mortensen, Henrik Skov Midtiby, R Nyholm Jørgensen, et al. Pixel-wise classification of weeds and crops in images by using a fully convolutional neural network. In *Proceedings of the International Conference on Agricultural Engineering, Aarhus, Denmark*, pages 26–29, 2016.

[19] Haozhang Gao, Mingyang Qi, Baoxia Du, Shuang Yang, Han Li, Tete Wang, Wenyu Zhong, and You Tang. An accurate semantic segmentation model for bean seedlings and weeds identification based on improved erfnet. *Scientific Reports*, 14(1):12288, 2024.

[20] Nikita Genze, Wouter K Vahl, Jennifer Groth, Maximilian Wirth, Michael Grieb, and Dominik G Grimm. Manually annotated and curated dataset of diverse weed species in maize and sorghum for computer vision. *Scientific Data*, 11(1):109, 2024.

[21] Raji Ghawi and Jürgen Pfeffer. Efficient hyperparameter tuning with grid search for text categorization using knn approach with bm25 similarity. *Open Computer Science*, 9(1):160–180, 2019.

[22] Thomas Mosgaard Giselsson, Rasmus Nyholm Jørgensen, Peter Kryger Jensen, Mads Dyrmann, and Henrik Skov Midtiby. A public image database for benchmark of plant seedling classification algorithms. *arXiv preprint arXiv:1711.05458*, 2017.

[23] Sanjay Kumar Gupta, Shivam Kumar Yadav, Sanjay Kumar Soni, Udai Shanker, and Pradeep Kumar Singh. Multiclass weed identification using semantic segmentation: An automated approach for precision agriculture. *Ecological Informatics*, 78:102366, 2023.

[24] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 2007.

[25] Nico Heider, Lorenz Gunreben, Sebastian Zürner, and Martin Schieck. A survey of datasets for computer vision in agriculture. *arXiv preprint arXiv:2502.16950*, 2025.

[26] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE access*, 8:4806–4813, 2019.

[27] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[28] Shan Huang, Ye He, and Xiao-an Chen. M-yolo: a nighttime vehicle detection method combining mobilenet v2 and yolo v3. In *Journal of Physics: Conference Series*, volume 1883, page 012094. IOP Publishing, 2021.

[29] Lamin L Janneh, Yongjun Zhang, Zhongwei Cui, and Yitong Yang. Multi-level feature re-weighted fusion for the semantic segmentation of crops and weeds. *Journal of King Saud University-Computer and Information Sciences*, 35(6):101545, 2023.

[30] Yu Jiang, Changying Li, Andrew H Paterson, and Jon S Robertson. Deepseedling: Deep convolutional network and kalman filter for plant seedling detection and counting in the field. *Plant methods*, 15(1):141, 2019.

[31] Priyank Kalgaonkar and Mohamed El-Sharkawy. An improved lightweight network using attentive feature aggregation for object detection in autonomous driving. *Journal of Low Power Electronics and Applications*, 13(3):49, 2023.

[32] Teja Kattenborn, Jana Eichel, and Fabian Ewald Fassnacht. Convolutional neural networks enable efficient, accurate and fine-grained segmentation of plant species and communities from high-resolution uav imagery. *Scientific reports*, 9(1):17656, 2019.

[33] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[35] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[36] Florian Kitzler, Norbert Barta, Reinhard W Neugschwandtner, Andreas Gronauer, and Viktoria Motsch. We3ds: An rgb-d image dataset for semantic segmentation in agriculture. *Sensors*, 23(5):2713, 2023.

[37] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[39] Jonathan M Levine, Peter B Adler, and Stephanie G Yelenik. A meta-analysis of biotic resistance to exotic plant invasions. *Ecology letters*, 7(10):975–989, 2004.

[40] Jiajia Li, Raju Thada Magar, Dong Chen, Feng Lin, Dechun Wang, Xiang Yin, Weichao Zhuang, and Zhaojian Li. Soybeannet: Transformer-based convolutional neural network for soybean pod counting from unmanned aerial vehicle (uav) images. *Computers and Electronics in Agriculture*, 220:108861, 2024.

[41] Yiting Li, Haisong Huang, Qingsheng Xie, Liguo Yao, and Qipeng Chen. Research on a surface defect detection algorithm based on mobilenet-ssd. *Applied Sciences*, 8(9):1678, 2018.

[42] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[43] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[44] Philipp Lottes, Raghav Khanna, Johannes Pfeifer, Roland Siegwart, and Cyrill Stachniss. Uav-based crop and weed classification for smart farming. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3024–3031. IEEE, 2017.

[45] Leo Prasanth Lourdu Antony. Rice leaf diseases dataset. *Mendeley Data*, 2023.

[46] Simon Madec, Kamran Irfan, Kaaviya Velumani, Frederic Baret, Etienne David, Gaetan Daubige, Lucas Bernigaud Samatan, Mario Serouart, Daniel Smith, Chrisbin James, et al. Vegann, vegetation annotation of multi-crop rgb images acquired under diverse conditions for segmentation. *Scientific Data*, 10(1):302, 2023.

[47] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International conference on Machine learning*, pages 23803–23828. pmlr, 2023.

[48] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[49] Andres Milioto, Philipp Lottes, and Cyrill Stachniss. Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2229–2235. IEEE, 2018.

[50] Sebastian Munz and David Reiser. Approach for image-based semantic segmentation of canopy cover in pea–oat intercropping. *Agriculture*, 10(8):354, 2020.

[51] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[52] Ayako Nakada, Ryota Niikura, Keita Otani, Yusuke Kurose, Yoshito Hayashi, Kazuya Kitamura, Hiroyoshi Nakanishi, Seiji Kawano, Testuya Honda, Kenkei Hasatani, et al. Improved object detection artificial intelligence using the revised retinanet model for the automatic detection of ulcerations, vascular lesions, and tumors in wireless capsule endoscopy. *Biomedicines*, 11(3):942, 2023.

[53] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[54] Alex Olsen, Dmitry A Konovalov, Bronson Philippa, Peter Ridd, Jake C Wood, Jamie Johns, Wesley Banks, Benjamin Girgenti, Owen Kenny, James Whinney, et al. Deepweeds: A multiclass weed species image dataset for deep learning. *Scientific reports*, 9(1):2058, 2019.

[55] Andres Patrignani and Tyson E Ochsner. Canopeo: A powerful new tool for measuring fractional green canopy cover. *Agronomy journal*, 107(6):2312–2320, 2015.

[56] Petchiammal, Briskline Kiruba, Murugan, and Pandarasamy Arjunan. Paddy doctor: A visual image dataset for automated paddy disease classification and benchmarking. In *Proceedings of the 6th Joint International Conference on Data Science & Management of Data (10th ACM IKDD CODS and 28th COMAD)*, pages 203–207, 2023.

[57] Artzai Picon, Miguel G San-Emeterio, Arantza Bereciartua-Perez, Christian Klukas, Till Eggers, and Ramon Navarra-Mestre. Deep learning-based segmentation of multiple species of weeds and corn crop using synthetic and real image datasets. *Computers and Electronics in Agriculture*, 194:106719, 2022.

[58] Roshni Polly and E Anna Devi. Semantic segmentation for plant leaf disease classification and damage detection: A deep learning approach. *Smart Agricultural Technology*, 9:100526, 2024.

[59] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, 20(53):1–32, 2019.

[60] Arpan Singh Rajput, Shailja Shukla, and SS Thakur. Soynet: A high-resolution indian soybean image dataset for leaf disease classification. *Data in Brief*, 49:109447, 2023.

[61] Manjot Rani and Munish Kumar. Mobilenet for human activity recognition in smart surveillance using transfer learning. *Neural Computing and Applications*, 37(5):3907–3924, 2025.

[62] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[63] Gianmarco Roggiolani, Julius Rückin, Marija Popović, Jens Behley, and Cyrill Stachniss. Unsupervised semantic label generation in agricultural fields. *Frontiers in Robotics and AI*, 12:1548143, 2025.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[65] Inkyu Sa, Marija Popović, Raghav Khanna, Zetao Chen, Philipp Lottes, Frank Liebisch, Juan Nieto, Cyrill Stachniss, Achim Walter, and Roland Siegwart. Weedmap: A large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming. *Remote Sensing*, 10(9):1423, 2018.

[66] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[67] Sovi Guillaume Sodjinou, Vahid Mohammadi, Amadou Tidjani Sanda Mahama, and Pierre Gouton. A deep semantic segmentation-based algorithm to segment crops and weeds in agronomic color images. *information processing in agriculture*, 9(3):355–364, 2022.

[68] Daniel Steininger, Andreas Trondl, Gerardus Croonen, Julia Simon, and Verena Widhalm. The cropandweed dataset: A multi-modal learning approach for efficient crop and weed manipulation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3729–3738, 2023.

[69] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *International Workshop on Deep Learning in Medical Image Analysis*, pages 240–248. Springer, 2017.

[70] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2820–2828, 2019.

[71] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[72] Luke Taylor and Geoff Nitschke. Improving deep learning with generic data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI)*, pages 1542–1547. IEEE, 2018.

[73] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 competition and demonstration track*, pages 3–26. PMLR, 2021.

[74] Jan Weyler, Federico Magistri, Elias Marks, Yue Linn Chong, Matteo Sodano, Gianmarco Roggiolani, Nived Chebrolu, Cyrill Stachniss, and Jens Behley. Phenobench: A large dataset and benchmarks for semantic image interpretation in the agricultural domain. *IEEE transactions on pattern analysis and machine intelligence*, 46(12):9583–9594, 2024.

[75] Qiangli Yang, Yong Ye, Lichuan Gu, and Yuting Wu. Msfca-net: A multi-scale feature convolutional attention network for segmenting crops and weeds in the field. *Agriculture*, 13(6):1176, 2023.

[76] Momchil Yordanov, Raphaël d'Andrimont, Laura Martinez-Sanchez, Guido Lemoine, Dominique Fasbender, and Marijn Van der Velde. Crop identification using deep learning on lucas crop cover photos. *Sensors*, 23(14):6298, 2023.

[77] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *International workshop on deep learning in medical image analysis*, pages 3–11. Springer, 2018.