INTERINSTITUTIONAL POST-GRADUATE PROGRAM IN SCIENCE AND TECHNOLOGY

CENTRO DE INVESTIGACIONES EN ÓPTICA A.C.

RGB AND MULTISPECTRAL IMAGE ANALYSIS BASED ON DEEP LEARNING FOR REAL-TIME DETECTION AND CONTROL OF WEEDS IN CORNFIELDS

THESIS

TO OBTAIN THE ACADEMIC DEGREE OF:

DOCTOR IN SCIENCE AND TECHNOLOGY IN THE SPECIALTY OF MECHATRONICS AND MECHANICAL DESIGN

PRESENTED BY:

M.Sc. Francisco Garibaldi Márquez

SUPERVISOR: **Dr. Luis Manuel Valentín Coronado**
CO-SUPERVISOR: **Dr. Gerardo Ramón Flores Colunga**

AGUASCALIENTES, AGUASCALIENTES.
APRIL 23, 2024

Abstract

Weeds drastically reduce the harvest volume of maize and the quality of forage if not controlled in time. Spraying herbicides is the most commonly used method for controlling weeds worldwide; however, it has led to environmental pollution. Intelligent mechatronic systems for mechanically removing weeds or selectively spraying herbicides are considered as alternatives to address the struggles associated with herbicide usage. Nevertheless, detecting undesired plants under authentic cornfields poses a significant challenge. Therefore, in this thesis, a vision system based on deep learning (DL) was proposed and developed for real-time detection and control of weeds in these complex scenarios. To develop the vision system, a large dataset of RGB and multispectral images was primarily created and annotated at the pixel level. Subsequently, both shallow and deep learning classification algorithms were explored. Additionally, End-to-end semantic segmentation convolutional neural networks (CNNs) were proposed for weed detection. The combined use of segmentation and classification networks was beneficial for detecting weeds in natural cornfields. Transformer architectures for semantic segmentation were also explored, yielding better results than CNNs. Our optimized transformer achieved a dice similarity coefficient (DSC) of 90.24 % and a mean intersection over union (mIoU) of 82.91 %. Afterward, a mechatronic platform commanded by the vision system, named the Smart Weed Sprayer (SWS), was developed. The SWS was evaluated in an authentic cornfield experiment, achieving a 45.64 % reduction in herbicide usage compared to a conventional weed sprayer (CWS). Moreover, similar effectiveness in weed control was observed between the SWS and CWS. Therefore, employing DL for weed control presents an alternative to reduce herbicide usage and production costs.

Keywords: deep learning, weed detection, weed control, smart weed sprayer, reduction of herbicide usage.

This thesis is dedicated to my children,
Maya Garibaldi Azcona and
Francisco Garibaldi Azcona,
and to my wife Diana L. Azcona Cózar.

Acknowledgments

I want to thank the people of Mexico, whose tax contributions enabled the Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCYT) to grant me the scholarship 299632, which made it possible for me to afford my doctoral studies.

I am very grateful to my thesis advisors Dr. Luis M. Valentín Coronado and Dr. Gerardo Flores for sharing their knowledge and encouraging me to successfully complete this important research project. Likewise, I thank my tutorial committee members, Dr. Diego A. Mercado Ravell and Dr. Alfonso Ramírez Pedraza, for their significant contributions made each four-month period to this research. Additionally, I want to thank M.C. Juan Margarito Sarabia Torres for his support in the instrumentation of the smart weed sprayer.

I would also like to express my gratitude to the CIO for opening the doors that allowed me to achieve my professional goal of completing a doctorate.

I want to thank the INIFAP, the institution to which I am affiliated, for giving me the opportunity to advance professionally and pursue my doctoral studies.

Thank you all.

Contents

List of figures

List of tables

Nomenclature

AI	Artificial Intelligence
ANN	Artificial Neural Network
$ANNs$	Artificial Neural Networks
BLW	Broad-leaf Weeds
CCA	Connected Component Analysis
CNN	Convolutional Neural Network
$CNNs$	Convolutional Neural Networks
CWS	Conventional Weed Sprayer
DC	Depthwise Convolutions
DL	Deep Learning
DSC	Dice Similarity Coefficient
$DSConv$	Depthwise Separable Convolutions
$Fast\,R-CNN$	Fast Region-based Convolutional Neural Network
$Faster\,R-CNN$	Faster Region-based Convolutional Neural Network
FCL	Fully Connected Layers
$FCNN$	Fully Convolutional Neural Network
$FCNs$	Fully Connected Networks
FN	False Negatives

FP	False Positives
GPU	Graphical Processing Unit
HSV	Hue Saturation and Value
HT	Hue Transform
IoU	Intersection over Union
KNN	K-Nearest Neighbors
LBP	Local Binary Pattern
$LBP_{P,R}^{riu2}$	rotation-invariant uniform Local Binary Pattern
$Mask\,R-CNN-ASPP$	Mask Region based Convolutional Neural Network with Atrous Spatial Pyramid Pooling
$Mask\,R-CNN$	Mask Region based Convolutional Neural Network
$mIoU$	mean Intersection over Union
ML	Machine Learning
MLP	Multilayer Perceptron
$NDVI$	Normalized Difference Vegetation Index
NLW	Narrow-leaf Weeds
PC	Pointwise Convolution
$R-CNN$	Region-based Convolutional Neural Network
RGB	Red, Green, and Blue channels
ROI	Region of Interest
$ROIs$	Regions of Interest
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
$SSWM$	Site Specific Weed Management

SVM	Support Vector Machine
SWS	Smart Weed Sprayer
TN	True Negatives
TP	True Positives
UAV	Unmanned Aerial Vehicle
ViT	Vision Transformer

Chapter 1

Introduction

The exigencies for food are growing every year along with the demand for fibers and biomass energies (Rabab et al., 2021); factors directly related to the agriculture sector. By 2050, these supply demands will rise by 70 % about what is needed currently because the human population is projected to be more than nine billion in that year (Monteiro and Santos, 2022). Regarding the corn crop (*Zea mays L.*), the third most cultivated cereal in the world, after rice (*Oryza sativa L.*) and wheat (*Triticum aestivum L.*), the kernel is transformed into a vast sort of food to feed humans. Therefore, the demand for this cereal will leap from $1,482.1Mt$ to around $2,521.1Mt$ globally (FAO, 2023) for the year above. Modern technology from diverse engineering disciplines should work together to ensure future food security by increasing crop yield per land area.

Nevertheless, increasing the harvest volume of any crop is challenging, not only due to the correct and opportune implementation of management practices but also due to factors such as weather change, which often bring long-drought periods year after year (Konduri et al., 2020). The resistance that certain pathogen insects and weed plants have gained to some chemical active ingredients (Westwood et al., 2018), which also complicates boosting crop yield. Among the problems above, weeds may be one of the most dangerous phenomena because they negatively affect the yield and quality of harvests Monteiro and Santos (2022). The reason is that weeds compete with crop plants for nutrients, sunlight, and water, and also they host-pathogen insects (Picon et al., 2022; Raja et al., 2020), that eventually transmit diseases to crop plants. Therefore, opportune weed eradication is a mandatory cultural practice, especially at early growth stages, for preventing competition among weeds and crop plants for resources (Garibaldi-Márquez et al., 2022).

Although corn is a noble and resistant crop that can be grown in various soil types and weather conditions, its yield is also affected by weed plants. Early work found a directly proportional relationship between the reduction of kernel yield and the increment of the biomass of weed plants (Yeganehpoor, 2015). In other work, Gao et al. (2018) registered a kernel loss of 29 % due to the competition of corn plants with weeds. Even worse, it has been documented that weeds can reduce the grain volume of this cereal by up to 90 % by surface area if weeds are never controlled (Knežević et al., 2021). In general, an estimated 29 % of this food is lost worldwide due to weeds (Gao et al., 2018).

The most common weed control methods are manual, mechanical, and chemical. Through manual control, it is possible to eliminate weeds in the row and crop-plant lines. However, this method is barely adequate for subsistence production systems (Wang et al., 2019). On the other hand, the mechanical method is suitable for intensive production systems due to the development of weeding and cultivator implements. Nonetheless, these types of equipment hardly eliminate 50 % of the weeds (Sabzi et al., 2020) because they are not able to handle the herbs located in the crop-plant lines (Van Der Weide et al., 2008). For this reason, the most often used method to eliminate weeds globally is by spraying herbicides (Hamuda et al., 2016), attributable to its effectiveness and practicality. According to Wang et al. (2018), the chemical method can eliminate between 90 to 99 % of interrow and intra-row weeds. However, they are spilled uniformly throughout the fields, even in regions where there are no herbs (Gao et al., 2018). Overusing these chemicals has produced water, soil, and air pollution (Monteiro and Santos, 2022; Quan et al., 2021). In this regard, the European Food Safety Authority (EFSA) documented that 98.9 % of agricultural food products contained specific residues of agrochemical molecules (Partel et al., 2019). Considering this problem, some European countries have legislated to achieve a gradual reduction or even a total prohibition of herbicides in their territories (Hamuda et al., 2016). This action has also been replicated in other countries such as Mexico, where the government has banned the *glyphosate* active ingredient. In substitution, producers have being encouraged to use environment-friendly herbicides from biological formulations (Alcántara-de la Cruz et al., 2021).

Stopping the use of synthetic herbicides completely in the short term is very difficult because farmers consider the method of spraying practical (Imoloame, 2017). Therefore, to mitigate the harm of these products, alternatives for the near future could rely upon the development of intelligent systems to spray weeds just where they are situated, a technique called Site-Specific Weed Management (SSWM) (Kamath et al., 2022).

The advances in computer and electronics science and engineering have permitted the development of computers with more data processing capacity. This has motivated the development of vision systems algorithms for mapping weeds and implementing them in practically real-time (Janneh et al., 2023). Preliminary studies have stated that SSWM could save from 45 to 66 % of herbicides without crop yield reduction, compared to those traditional methods of uniform application (Christensen et al., 2003; Monteiro and Santos, 2022). Recently, Nikolić et al. (2021) reported up to 82 % of herbicide reduction when SSWM and time-specific weed control (TSWC) were combined. However, the practical implementation of SSWM in natural fields presents a significant challenge regarding weed-crop discrimination. Given the excessive variability inherent to such environments, most vision algorithms struggle with accurately localizing and classifying plant species.

In this thesis, an intelligent vision system based on DL models will be developed and investigated for detecting weed plants within natural cornfields. Subsequently, the vision system will be installed over a mechatronic platform to conform a smart weed sprayer (SWS). This SWS will be able to navigate through cornfields pulled by an agricultural tractor spraying weeds in real-time.

The remainder of this chapter is organized as follows. Section 1.1 gathers the most relevant works for weed classification, detection, and segmentation by using either classical algorithms, machine learning (ML), or DL. The justification of the research is provided in Section 1.2, whereas Section 1.3 lists the objectives of the research. Finally, Section 1.4 states the hypothesis of the work.

1.1. Background

The first challenge to overcome when implementing an automatic weed control technique is discriminating weeds from crop plants (Wang et al., 2019). The used algorithms should first extract the relevant features of crop plants, weeds, soil, and residues. Then, these algorithms must be able to discriminate each of these elements.

This section presents spectroscopy reflectance characteristics, image processing-based algorithms, and DL-based architectures for crop and weed discrimination.

1.1.1. Reflecting characteristics for crop/weed discrimination

Spectral reflectance has been used to distinguish crops and weeds. Plant leaves contain different concentrations of light absorbent compounds, such as α-carotenoid (420, 440, and $470nm$), anthocyanin ($400-550nm$), β-carotenoid (425, 450, and $480nm$), chlorophyll a (435, $670-680$, and $740nm$), chlorophyll b (480 and $650nm$), lutein (425, 445, and $475nm$), moisture (970, 1450, and $1944nm$), and violaxanthin (425, 450, and $475nm$) (Zwiggelaar, 1998). These compounds provoke peculiar reflectance curves among plant species. As an illustration, Figure 1.1, which has been adapted from the work of Kyllo (2003), provides reflectance curve behaviors of eight crops, wheat stubble, and soil. As it is observed, the reflectance drastically increases in the RedEdge region ($670-760nm$), the transition from the red region to the NIR region. Therefore, the RedEdge region has been widely used by scholars to discriminate among plant species, whereas the near-infrared (NIR) region ($700-1400nm$), in which the reflective differences between plants, soil, and residues are pronounced, is commonly employed to discriminate vegetation from the background (Steward et al., 2019).

Many works have applied spectral reflectance for crop and weed discrimination using VIS/NIR/MIR spectroscopy. Vrindts et al. (2002) discriminated sugarbeet/weeds and corn/weeds, including seven weed species, based on VIS/NIR ($480-820nm$), reporting over 90 % of correct classification. The discrimination

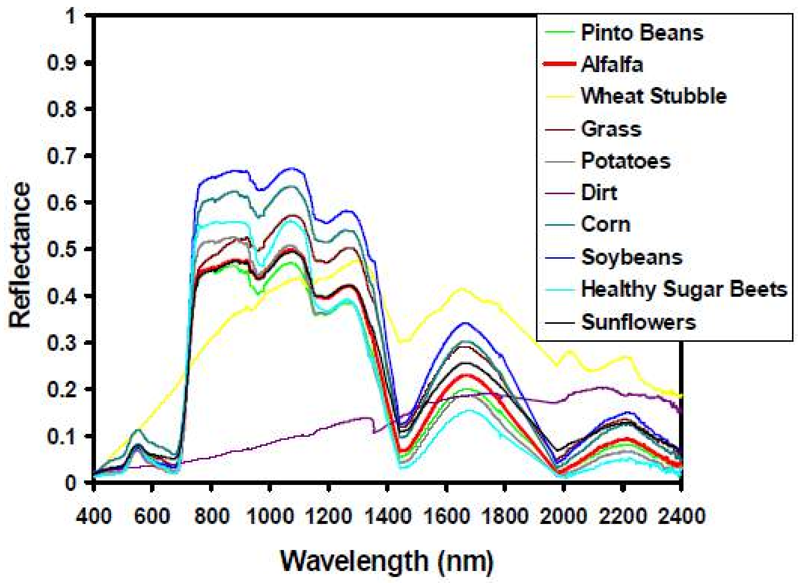


Figure 1.1: Spectral reflectance of some crops, soil, and stubble. Adapted from Kyllo (2003).

of corn plants from weeds in open fields at early growth seasons using hyperspectral data in the VIS/NIR band ($408 - 947mn$) based on Suppor Vector Machine (SVM) and Artificial Neural Network (ANN) classifiers have also been explored, achieving accuracy values of 69.2 % and 58.3 %, respectively (Karimi et al., 2006). In other work, Pantazi et al. (2016) discriminated corn plants and weeds using spectral reflectance and the classifiers auto-encoder network, SVM, one-class mixture of Gaussians (MOG), and one-class self-organizing map (SOM). In this work, the authors indicated that the classifier MOG and SOM performed better than the rest of the classifiers; corn was 100 % classified as such by MOG and SOM, whereas weeds were classified in the range of 31 %−98 % by MOG and 53 %-94 % by SOM. In the work of Che'Ya (2016), a trial for discriminating *Sorghum* crops from eight weed species through spectral reflectance (VIS/NIR) was carried out, obtaining accuracy values between 85 % and 100 %. Finally, Pott et al. (2020) evaluated the accuracy of spectral bands to discriminate solely weeds from crop mulch and soil in pre-planting conditions. Other carried out works for weed discrimination using spectroscopy are listed in Table 1.1.

Table 1.1: Works over crop plants and weeds discrimination using spectral reflectance.

Crop	Weed species	Wavelength range used (nm)	Model	Accuracy (%)	Reference
Soybean	*Acalypha australis L.* *Marsilea quadrafolia L.*	400 – 670, 680 – 700, 720 – 1000	PLS	98.3 %	Zhang and He (2006)
Soybean	*Eleusine indica L.* *Alternanthera philoxeroides* *Amaranthus viridis L.*	VIS/NIR	ANN	100 %	Zhu et al. (2008)
Corn	*Beta vulgaris L.*	635,685,785	SVM	97 %	Akbarzadeh et al. (2018)
Corn	*Dchinochloa crasgalli* *Echinochloa crusgalli*	VIS/NIR	SVM, ANN, DT	84.21 %	Deng et al. (2014)
Corn	*Digitaria ischaemum (Schreb.) I* *Echinochloa crus-galli (L.) Beauv.* *Panicum capillare (L.)* *Setaria glauca (L.) Beauv.* *Ambrosia artemisiifolia (L.)* *Amaranthus retroflexus (L.)* *Chenopodium album (L.)* *Capsella bursa-pastoris (L.) Med.*	400 – 425, 425 – 490	PLSDA	94.8 %	Panneton et al. (2010)
Corn, Barley, Wheat, Sugar beet	*Echinochloacrus-galli L.* *Avenafatua L.* *Alopecurusmyosuroides* *Chenopodium album L.*	VIS/NIR	CA	100 %	Meinen and Rauber (2015)
Sugarcane	*Commelina benghalensis* *Brachiaria brizantha* *Brachiaria decumbens* *Panicum maximum cv* *Alternanthera tenella* *Ipomoea hederifolia* *Ipomoea purpurea* *Ricinus communis L.* *Ageratum conyzoides* *Crotalaria juncea* *Stizolobium aterrimum*	500 – 550, 650 – 750, 1300 – 1450, 1800 – 1900	SIMCA, RF	97 %	de Souza et al. (2020)
Wheat, Chickpea	*Echinochloacrus-galli L.* *Setaria viridis* *Eleusine indica L.* *Digitaria* *Chenopodium quinoa*	567, 667, 715, 1345, 1402,1725, 1925, 2015	Bayesian	84.3 %	Deng et al. (2016)

PLS–partial least squares; ANN–artificial neural network; SVM–support vector machines; DT–decision tree; PLSDA–partial least square discriminant analysis; CA–cluster analysis; SIMCA–soft independent modeling of class analogy; RF–random forest.

Although the works above have obtained acceptable results on the discrimination of crops and weeds in natural fields, they have been performed to ensure that the reflectance signal is not affected by external sunlight intensities. In this regard, some researchers have concluded that the discrimination of crop plants from weeds using spectrometers is complicated due to the influence of sunlight. Additionally, crops and weeds have similar reflections, especially at early growth stages (Pérez-Ortiz et al., 2015), and the reflection changes with the growth stage of the plants. This method is adequate only for a laboratory scale, in which the environmental conditions can be controlled. Optoelectronic sensors, which measure reflection intensities usually in the spectrum's red/near-infrared (R/NIR) region, have been seen as an alternative. However, this kind of sensor can discriminate vegetation (crops and weeds) exclusively from the background (soil and residues) (Biller, 1998). Therefore, it is barely applied to crops that follow a sown pattern in clearly separated plant lines to detect weeds in the rows. Nonetheless, when crop plants are close to each other or when crops have been broadcasting sown, the systems fail.

1.1.2. Classic image processing algorithms for crop/weed discrimination

Proximal sensors based on cameras in machine vision systems have been commonly studied for crop and weed distinction. The typical procedure includes preprocessing, segmentation, feature extraction, and classification (Liu and Bruch, 2020; weis and Sökefeld, 2010), as shown in Figure 1.2.

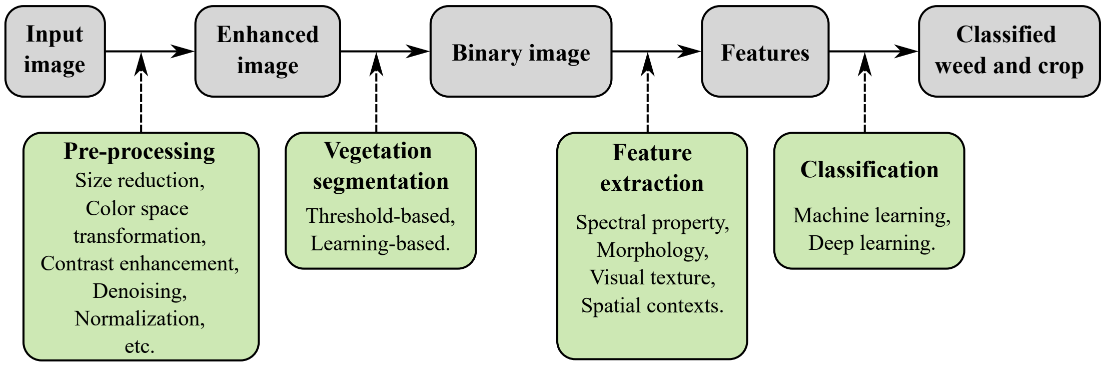


Figure 1.2: Common workflow of image processing-based weed recognition.

Pre-processing

Pre-processing images for weed detection usually start by reducing the resolution of images, to reduce the computation cost. Subsequently, color space transformation is done principally to eliminate soil pixels later and leave just vegetation pixels in input images. Since RGB color space is often used for visualization but not suitable for segmentation and analysis due to the high correlation between Red (R), Green (G), and Blue (B) channels, it is almost mandatory to perform a color transformation. The color space HSV (hue, saturation, and value) is often used for crop/weed separation (Garibaldi-Márquez et al., 2022). According to Hamuda et al. (2016), in this color space the channels are not correlated, for that reason is good for outdoor segmentation since the illumination value is correlated with the S channel. Other color spaces studied for vegetation separation from the soil are HSI (hue, saturation, and intensity) (Li et al., 2016), Lab (L for illumination, a for values from red to green, and b for values from blue to yellow) (Chen et al., 2021) and $YCrCb$ (Y for luminance component, Cb for blue-difference chroma component, and Cr for red-difference chroma component) (Wang et al., 2020).

Noise and contrast of images captured in outdoor field conditions are seriously altered by the weather conditions (sunny, cloudy, rainy, etc.) and capture time of the day (morning, noon, afternoon, etc.), making the greenness identification complicated (Yang et al., 2015). Pulido-Rojas et al. (2016a) used median filtering for noise suppression as a pre-processing step for detecting multiple weed plants in open fields. Additionally, Rakhmatulin (2020) described the implementation of blur, Gaussian, bilateral, and Laplacian filtering for image denoising on tasks of weed recognition. On the other hand, adjusting the gray level in the range $[0-255]$ has been used to enhance contrast for plants and soil separation (Liu et al., 2014). In another work, Siddiqi et al. (2014) improved the contrast and light intensities of images for greenness identification by equalizing their global histogram. As a final image pre-processing step, normalization is a typical action performed in which the range of pixel values of the input images is changed to a new one that is more familiar or normal to the senses (Wang et al., 2019).

Vegetation segmentation

Removing the soil and other residues, such as mulch and stone pixels, from vegetation is a typical subsequent process implemented in weed detection. Although in Figure 1.2 learning based is pointed out, in this section, solely vegetation segmentation based on thresholding is covered. Some works have segmented vegetation straight from the color spaces HSV, HSI, Lab or $YCrCb$. Such as in the work of Yang et al. (2015), in which the maize seedlings were separated from the background elements: soil, straw ash, plastic film, corn straw, and wheat straw, by thresholding the hue value of images. Chen et al. (2021) separated green vegetation (corn and weed plants) from the soil in natural conditions in the *Lab* color space. Other scholars have explored the transformation of the images to color indices derived from the color space RGB, the normalized *rgb*, or even indices derived from multispectral cameras for vegetation and soil separation. In this sense, in the work of Liu et al. (2020) maize plants were detected in natural conditions, using the color indices excess green (*ExG*), excess red (*ExR*) and *ExG* minus *ExR* and thresholding the images by Otsu method (Otsu, 1979). The NDVI index is the most widely used for this task, which is derived from multispectral images. Table 1.2 provides a list of common color-based indices for vegetation segmentation reported in the literature.

Table 1.2: Typical color-based indices for vegetation segmentation.

Index	**Color space**	**Formula**	**Reference**
Excess Green Index (*ExG*)	*rgb*	$ExG = 2g - r - b$	JIN et al. (2021); Liu et al. (2020); Yang et al. (2015) Espejo-Garcia et al. (2020); Guerrero et al. (2012)
Excessive Green (*EG*)	*RGB*	$EG = 2G - R - B + 127$	Mathanker et al. (2010)
Excess Red Index (*ExR*)	*RGB*	$ExR = 1.3R - G$	Liu et al. (2020); Milioto et al. (2018); Wang et al. (2020)
Modified Excess Green Index (*MExG*)	*RGB*	$MExG = 1.262G - 0.884R - 0.311B$	Burgos-Artizzu et al. (2011); Wang et al. (2020)
Modified Excess Green Index 1 (*MExG1*)	*RGB*	$MExG1 = 2G - R - B$	Wu et al. (2011)
Normalized Excessive Green (*NEG*)	*RGB*	$NEG = 2.8G - R - B$	Jeon et al. (2011); Karadöl et al. (2020)
Green minus Red (*GMR*)	*RGB*	$GMR = G - R$	Bakhshipour and Jafari (2018)
Green Pixel Count (*GPC*)	*RGB*	$GPC = 2G - R * G - B$	Prema and Murugan (2016)
Excess Green minus Excess Red Index (*ExGR*)	*RGB*	$ExGR = ExG - ExR$	Le et al. (2019); Wang et al. (2020); Yang et al. (2015)
Color Index of Vegetation Extraction (*CIVE*)	*RGB*	$CIVE = 0.441R - 0.811G + 0.385G + 18.78745$	Milioto et al. (2018); Wang et al. (2020); Yang et al. (2015)
Vegetative Index (*VEG*)	*RGB*	$VEG = \frac{G}{R^{0.667}B^{0.333}}$	Wang et al. (2020); Yang et al. (2015)
Combined Indices (*COM*)	*RGB*	$COM = ExG + CIVE + ExGR + VEG$	Wang et al. (2020); Yang et al. (2015)
Combined Indices 1 (*COM1*)	*RGB*	$COM1 = 0.36ExG + 0.47CIVE + 0.17VEG$	Guerrero et al. (2012); Wang et al. (2020)
Normalized Difference Index (*NDI*)	*RGB*	$NDI = 128 * \left(\left(\frac{G-R}{G+R}\right) + 1\right)$	Lin et al. (2017); Milioto et al. (2018)
Normalized Green-Red Difference Index (*NGRDI*)	*RGB*	$NGRDI = \frac{G-R}{G+R}$	Barrero and Perdomo (2018); Hunt et al. (2005)
Normalized Difference Vegetation Index (*NDVI*)	*RGB-NIR*	$NDVI = \frac{NIR+R}{NIR-R}$	Barrero and Perdomo (2018); Lottes et al. (2016, 2017) Milioto et al. (2018); Potena et al. (2017)

Although color indices provided acceptable results for greenness identification, some scholars argue that they are directly affected by sunlight intensity; as a result, the regions of interest (ROIs) considered as vegetation often enclose pixels of soil or ROIs considered as soil include pixels of vegetation (Wang et al., 2019).

Feature extraction

Distinguishing crop plants from weed plants is the most difficult task for SSWM implementation. Most traditional image pre-processing methods employ feature differences among the plant leaves, including spectral properties, morphology, visual texture, and spatial contexts.

Spectral features for discrimination among plants are based on multispectral, hyperspectral, and spectral indices imagery. Nonetheless, spectral features are adequate solely when plant species to discriminate have distinct leaf colors. Such as in the work of Pgnatti et al. (2016), in which hyperspectral imagery was used to separate corn plants from five weed species by implementing the PROSAIL model. Herrmann et al. (2013) studied the ground-level spectroscopy imagery for detecting annual grasses and broadleaf weeds in wheat fields by using partial least squares discriminant analysis (PLSDA), obtaining a total accuracy of 85 %. The classification of corn crop and three weed species was done with the classifiers random forest (RF) and k-nearest neighbors (KNN) reaching a classification rate of 100 % for corn plants and a mean classification of 74.4 % for weeds for RF, which was better than KNN (Gao et al., 2018). In other work, Louargant et al. (2018) discriminate corn and sugarbeat crops from diverse weed species, extracting spectral information from four-band multispectral images and training an SVM model, obtaining a detection rate of 75 %. Carrot and three weed species were successfully discriminated by implementing an imaging spectrometer system by means of SVM and linear discriminant analysis (LDA). The SVM model reached 85 % and 90 % when the features of eight bands and 15 bands were extracted, respectively (Liu et al., 2019). The discrimination of two monocotyledon weed species in rice crops was also explored by Zhang et al. (2019b) using hyperspectral images. Particularly, in this work, the authors have reported that a 100 % and 92 % recognition rate for the weeds and rice was reached when six spectral features were used. However, in most cases, crop plants and weeds share the "green color", which results in no separation action, especially at early growing stages when crop and weed have similar reflectance characteristics (Peteinatos et al., 2013).

The shape and structure of any part of a plant, also known as biological morphology, have been used to identify plant species. Shape features are broadly divided into shape parameters, region-based descriptors, and contour-based descriptors (Wu et al., 2021). Shape parameters include perimeter, area, diameter, minor axis length, major axis length, eccentricity, compactness, rectangularity, circularity, convexity, and solidity of the segmented regions (Bakhshipour and Jafari, 2018; Herrera et al., 2014). These parameters are easy to implement and are not affected by sunlight intensity. As region-based descriptors, we found the Hu's moment invariants (MI) (Hu, 1962) and two-dimensional Fourier descriptors(FDs) (Pereira et al., 2012). Hu's MI is represented by seven MI parameters derived from the contour and the internal silhouette of the segmented regions. These features are independent of geometric translation, scaling, and rotation and are robust to noise. In the case of two-dimensional FDs, they measure the shape properties by establishing feature points in the region plane and carrying out Fourier transforms on rows and columns at the same time (Wu et al., 2021). Contour-based descriptors often include spatial position descriptors, curvature scale descriptors, and one-dimensional Fourier descriptors.

Although each category of shape features has the potential to distinguish among plant species by itself, in literature, they often are combined to obtain robust models. Chen et al. (2015) used eight shape features and seven Hu MI to detect soybean plants in open fields, while the rest of the plants were considered as weeds, obtaining more than 90 % of accuracy. In other work, Pereira et al. (2012) classified three aquatic weeds combining the shape descriptors Beam Angle Statistics (BAS), Fourier Descriptors (FD), Hu MI, Multiscale Fractal dimension (MS) and Tensor Scale Descriptor (TSD) obtaining a maximum recognition rate of 96.41 %. Corn plant discrimination had reached 100 % under field environment simulation in a laboratory when Hu MI and KNN were implemented (Midtiby et al., 2011). The discrimination between monocotyledons and dicotyledons weeds in natural corn fields have shown 92.9 % of accuracy when Hu MI and six geometric shape descriptors were classified with fuzzy multicriteria decision making, surpassing the performance of SVM, Choquet fuzzy integral, the Sugeno fuzzy integral and the Dempster-Shafer theory (Herrera et al., 2014).

On the other hand, texture features, which reflect the spatial distribution of pixels, have been reported lately to be efficient for discriminating crops from weeds. This is because leaves' veins differ in texture among species, and the roughness of leaves' surface also changes (Wu et al., 2021). The texture feature methods are principally divided into four categories: i) statistical features, ii) structural features, iii) model-based features, and iv) transformed-based features (Wang et al., 2019). On plant discrimination, into statistical feature category, the most frequently reported feature descriptors are Gray-level Co-occurrence Matrix (GLCM) (Haralick et al., 1973) and Gray-level Gradient Co-occurrence Matrix (GLGCM) (Lam, 1996). Bakhshipour et al. (2017) extracted 52 GLCM texture features in four directions from wavelet images for weed segmentation of sugarbeet crop using ANN as the classifier, obtaining 96 % of accuracy. In the category of structural features, the texture operator local binary pattern (LBP) (Ojala et al., 2002) is widely used because it is robust enough to monotonic grey-level transformation, scaling, viewpoint, illumination invariance, and rotation invariance (Hamouchene et al., 2014). In the work of Le et al. (2019), the crops Corn, Canola, and radish were discriminated using LBP descriptors and SVM, obtaining an accuracy of 91.85 %. In a distinct work, Le et al. (2020a) reported an algorithm named *filtered Local Binary Pattern with contour masks and coefficient "k"* (k-FLBPCM); here, the authors have implemented an SVM classifier, achieving an accuracy of 90.94 %. Finally, Gabor filters are more often used for the recognition of crops and weeds into the group of transformed-based features category. Nonetheless, it is always accompanied by other texture descriptors. Like in the work o Chaki et al. (2015), in which grayscale images were convolved with the Gabor filter, and then GLCM texture features were computed for leaves recognition.

The final feature extraction algorithms for crop and weed discrimination take advantage of the spatial contexts of the crop by locating the line of crop plants (Xu et al., 2020b). Then, the plants that were outside the line were considered weeds. As it could be inferred, they are more accurate for crops that follow a sown pattern and are not adequate for crops broadcasting sown. In order to detect crop rows, the Hough transform (HT) algorithm is frequently used, and the methods Harries corner, pixel histogram, edge, vertical projection, and linear scanning are also used. For instance, Tellaeche et al. (2008) identified isolated weeds located in corn rows by localizing the crop lines throughout HT. The Harries corner method was successfully used for detecting crop rows as a pre-step for estimating the density of weeds (Xu et al., 2020b). In another work, Asif et al. (2010) detected the edge of corn rows and then, with the help of the HT delimited the area between the crop lines for controlling weeds. In the work of Wu et al. (2011), the center line of wheat rows was located using the pixel histogram method, then the wide of the rows was delimited, finding the edge of the rows. Then, plants outside of that area were considered as weeds. Also, the weed area in corn crop was identified throughout the pixel histogram method for a variable rate spraying system (Xu et al., 2018). In the work of Tang et al. (2016), the vertical projection method and the linear scanning method were combined to detect corn plant lines for site-specific spraying of weeds.

Plants' features effectively discriminate crops and weeds under low plant-density scenarios (Wu et al., 2021). In this way, the following Table 1.3 provides the advantages and disadvantages of the aforementioned features.

Table 1.3: Advantages and disadvantages of common features for weed recognition.

Features	Advantages	Disadvantages
Spectral property	Spectral features are robust to partial occlusion of foliage.	Crop and weed plants share green color, especially at early growing stages, having similar reflectance characteristics.
Morphology	Easy to implement and are not affected by sunlight intensity.	Monocotyledon and dicotyledon crops and weeds share similar morphology at early grown stages, making the discrimination very difficult.
Visual texture	Most of the methods are robust to monotonic grey-level transformation, scaling, viewpoint, illumination invariance, and rotation invariance.	They do not meet the real-time requirements.
Spatial contexts	The sowing pattern of crops improve the discrimination accuracy at early growth stage of the weeds.	Weeds located in the line of crop plants are not distinguished. Also, under a high density of weeds, it is difficult to detect the crop rows. Finally, some algorithms fail to detect the crop rows at the border of the fields.

The recognition of crops and weeds in open fields is difficult using a single feature of the plants due to the interference of uncontrolled factors. Thus, many works have combined features to increase the discrimination potential. In this way, De Rainville et al. (2014) combined HT for crop row detection and morphological features for the classification of weeds from corn and soybean fields. They reported a classification average of 94 % for corn and soybean plants and 85 % for weeds. Pérez-Ortiz et al. (2015) mapped weeds in sunflower crops using vegetation index images, NDVI images, and detection of crop rows using HT. Then, the following classifiers were tested: unsupervised classifiers k-means and Repeated k-means, the semi-supervised SVM and the supervised KNN, linear SVM, and SVM. Finding that semi-supervised SVM obtained similar performance to that of supervised classifiers. Lottes et al. (2017) segmented weeds in sugarbeet crop implementing diverse image-based processing techniques. First, they segmented vegetation from the background using the NDVI index; then, images were transformed to the color space Hue Saturation and Lightness (HSL), from which statistical features, texture features using LBP, and shape features were extracted and combined. As a final example, Chen et al. (2021) studied the combination of multi-features for weed detection in corn fields to obtain better feature descriptor combinations. They work with the following feature descriptors: rotation-invariant LBP, HOG, GLCM, GGCM, Hu moment invariant, and Gabor. As a result, they reported that a combination of rotation-invariant LBP and GGCM showed the highest accuracy of 97.50 %.

1.1.2.1. Classification of crops and weeds

According to the works reported in the literature, obtaining an acceptable performance (when implementing classic ML methodologies after feature extraction) depends on the correct classifier selection. Conventional ML-based classifiers usually reported in literature for crop and weed species are SVM (Bakhshipour and Jafari, 2018; Chen et al., 2021; Le et al., 2019; Pulido et al., 2017), ANN (Dadashzadeh et al., 2020; Nikolić et al., 2021; Torres-Sánchez et al., 2021; Zhu et al., 2008), KNN (Dadashzadeh et al., 2020; Gao et al., 2018; Pulido-Rojas et al., 2016b), RF (Gao et al., 2018; Kamath et al., 2020; Lottes et al., 2016), Bayesian algorithm (Deng et al., 2014; Tang et al., 2016), Bayesian classifier (De Rainville et al., 2014), AdaBoost (dos Santos Ferreira et al., 2017; Xu et al., 2020a), and k-means (Chen et al., 2021; Zhang et al., 2019a). Other classifiers found are PLSDA (Herrmann et al., 2013), LDA (Liu et al., 2019), and fuzzy multicriteria decision (Herrera et al., 2014). The adaptation of the correct classifier depends on the complexity of the task, whether it is binary or multi-class; the amount of data also impacts the output performance. For instance, SVMs can address the challenges of nonlinear and high-dimensional pattern recognition. They also demonstrate strong performance in handling small-sample and non-local minimum problems (Wu et al., 2021), whereas ANN has a strong learning capability and can classify unseen data (Bakhshipour et al., 2017). Therefore, many scholars have iterated on classifiers to find the best that adapts to their datasets.

For instance, in the work of Pereira et al. (2012), the classifier SVM, Bayesian classifier, multilayer perceptron (MLP), Self-Organized Maps (SOM), and Optimum-Path Forest (OPF) were evaluated on classifying the aquatic weeds *E. crassipes*, *P. stratiotes* and *S. auriculata.* These models were trained with shape descriptors Beam Angle Statistics (BAS), Fourier Descriptors (FD), Hu MI, Multiscale Fractal dimension (MS), and Tensor Scale Descriptor (TSD). The authors found that the OPF classifier trained with BAS descriptors was the better option for recognizing the weeds. In other work, Bakhshipour and Jafari (2018) found that SVM exhibited an overall accuracy of 95 %, whereas ANN reached 92.92 % when they were trained with shape features of common weeds. The dataset comprises 600 images acquired in "real" field conditions. However, the sunlight was obstructed when images were acquired. In a binary approach to classifying vegetable crops and weeds, an SVM model better classified the class crop while a KNN model classified better the class weed (Pulido-Rojas et al., 2016b) when trained with GLCM texture features. In other research, a comparative analysis on classifying soybean crop, broadleaf, and grass weeds using the classifiers SVM, AdaBoost, and RF indicated that SVM was better than AdaBoost and RF when the dataset was balanced. Nevertheless, when the dataset was unbalanced, the Adaboost models surpassed the performance of the SVM and RF models. These models were trained with GLCM, HOG, and LBP descriptors, and the minimum, maximum, mean, and standard deviation attributes of each channel of RGB, HSV, and CIELab color spaces (dos Santos Ferreira et al., 2017).

1.1.3. Deep learning models for crop/weed discrimination

DL models for weed classification, detection, and segmentation typically employ Convolutional Neural Networks (CNNs). CNNs consist of convolutional layers and fully connected layers. The convolutional layers execute operations like convolutions between filters and input data, capturing spatial and temporal features. On the other hand, fully connected layers are responsible for classifying the features extracted by the convolutional layers. Networks with more than three layers are commonly referred to as deep networks.

The performance that the AlexNet architecture (Krizhevsky et al., 2012) achieved in classifying images from the ImageNet dataset (Deng et al., 2009) in the large scale visual recognition challenge in 2012 was a benchmark for diverse DL-based computer vision tasks. This superior performance also motivated scholars to duplicate efforts on the application of this CNN architecture and develop new ones for weed classification, detection, and segmentation. Nowadays, the commonly used DL architectures for weed identification in open fields include CNNs and fully connected networks (FCNs) (Kamilaris and Prenafeta-Boldú, 2018). Besides, since 2017, the self-attention modules proposed by Vaswani et al. (2017) in the transformer architecture have also been utilized for weed identification.

Deep learning-based architectures are deeper than traditional ML architectures. That is, they have more hidden layers that transform the input data using diverse operations, allowing the representation of the data in a hierarchical way (Hu et al., 2021). Then, what make CNNs, FCNs, and Transformer interesting is that they can extract and learn multiple features of the input data on their own during the training process, and then, they can discriminate new unseen data at relatively high performance in real close-time using GPUs (Moutik et al., 2023).

The three aforementioned DL architecture groups for weed identification use RGB, multispectral, and hyperspectral imagery. Other works have used color indices images derived from RGB or multispectral channels. The DL architectures must be trained with a large dataset to obtain a high performance and robustness to adapt to new information. Figure 1.3 gives the usually followed steps for training a DL architecture for weed recognition. It is worth mentioning that not all the sub-steps need to be implemented, principally on the classification of crop/weed tasks, but the steps of data acquisition, dataset preparation, image pre-processing, training DL architecture, and testing of the models are mandatory.

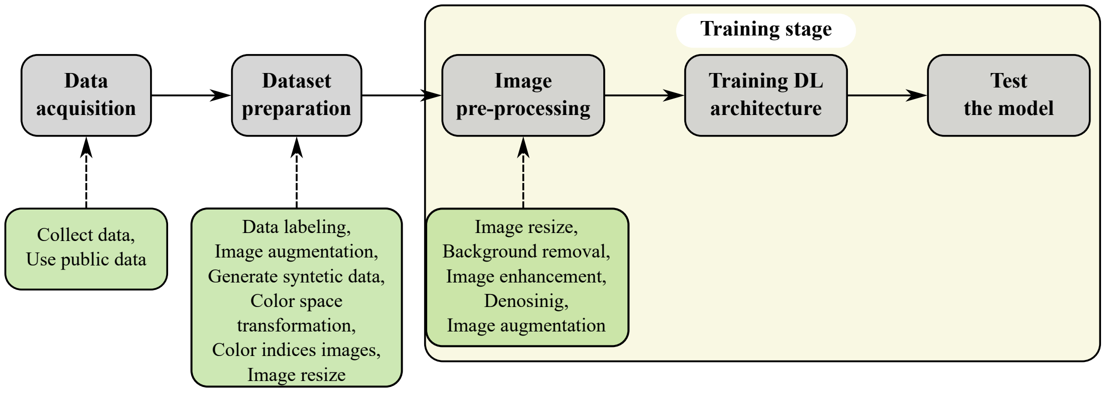

Training stage
Data acquisition
Dataset preparation
Image pre-processing
Training DL architecture
Test the model
Collect data, Use public data
Data labeling, Image augmentation, Generate syntetic data, Color space transformation, Color indices images, Image resize
Image resize, Background removal, Image enhancement, Denosinig, Image augmentation

Figure 1.3: Common DL workflow for weed recognition.

Data acquisition

For developing an automatic vision system for weed identification, the first step is to acquire the data. RGB, multispectral, or hyperspectral images could be acquired using Unmanned Aerial Vehicles (UAV), field robots, and all-terrain vehicles or captured manually. For instance, a drone DJI Mavic 2 Pro, equipped with an RGB camera, that flew five meters above the soil surface has been used to create an image dataset containing *sorghum* crop and diverse monocotyledon and dicotyledon weeds (Genze et al., 2022). In another study, Xu et al. (2023) also captured images using a DJI Phantom 4 V2 Pro drone equipped with an RGB camera, of soybean crops and multi-species of weeds. In this case, the drone was flown six meters in height. Respecting the use of field robots, Lottes et al. (2020) employed a robotic platform called “BoniRob” equipped with a 4-channel RGB+NIR camera for capturing images of sugarbeet and weeds. Meanwhile, Le et al. (2020a) mounted a multispectral camera over an all-terrain vehicle to generate a dataset of canola and radish crops. It is worth noting that many of the studies consulted used manually generated datasets for their experiments. In this way, Zhang et al. (2023) used an iPhone XS to generate a corn dataset of images captured at 60 centimeters. A different dataset of ten weeds that commonly grow in grasslands was acquired manually using smartphones (Jiang et al., 2023).

Other researchers have used publicly available datasets to create or complement their data to train their DL models. Few public datasets specialize in the classification, detection, and segmentation of crops and weeds. Table 1.4 lists some public datasets already annotated, which scholars have used for training DL models.

Data preparation

The dataset preparation step starts labeling the dataset according to the purpose of the research, which could be classification, detection, or segmentation. Datasets for training classification networks are integrated with images that have single plants, or at least one plant is predominant in the image. Then, the dataset is annotated at the image level. On the other hand, datasets dedicated to detecting crops and weeds could contain multi-plant images. The plants in the images are labeled individually by tracing a bounding box, which provides the localization and the class to whom each plant belongs. Finally, whether the dataset would be for segmentation, the images are annotated at the pixel level. Labeling images is a difficult and time-consuming task because images may contain multiple plant species, and each plant's pixels must be enclosed by tracing a polygon. Furthermore, if the image contains a high density of plants, the operation becomes more complicated due to the occlusion and overlap of the plants.

On the other hand, if the dataset size is small, a data augmentation technique must be used to obtain a robust model capable of performing correctly. Standard augmentation techniques for crop/weed identification are flipping, cropping, rotation, translation, and noise injection (Shorten and Khoshgoftaar, 2019). Other scholars create synthetic images to enlarge their dataset using DL architectures to retain similar features of the original instances while avoiding the annotation work. Such as in the work of Espejo-Garcia et al. (2021), in which plenty of synthetic images derived from the "early crop weed" dataset (Espejo-Garcia et al., 2020) were created using a generative adversarial network (GAN) for weed identification.

Table 1.4: Public available crop/weed datasets.

Dataset name	Crop	Weed species	Image type	Purpose	Reference
Soybean and weeds	Soybean	Grass and broadleaf weeds	RGB	Classification	dos Santos Ferreira et al. (2017)
Plant seedlings dataset	Corn, Sugar beet and Wheat	Nine weed species, broadleaf weeds and narrow-leaf weeds	RGB	Classification	Giselsson et al. (2017)
Leaf counting dataset	Not specified	Eighteen weed species	RGB	Classification	Teimouri et al. (2018)
DeepWeeds	Not specified	Eight weed species	RGB	Classification	Olsen et al. (2019)
Early crop weed	Tomato and cotton	*Solanum nigrum L.* and *Abutilon theophrasti Medik.*	RGB	Classification	Espejo-Garcia et al. (2020)
Corn, lettuce and weed	Corn and lettuce	*Cirsium setosum* *Chenopodium album* *Cyperus esculentus* *Poa annua*	RGB	Classification	Jiang et al. (2020)
Perennial ryegrass and weed	Perennial ryegrass	*Euphorbia maculata* *Glechoma hederacea* *Taraxacum officinale*	RGB	Classification and detection	Yu et al. (2019a)
Sugar beet dataset	Sugar beet	Nine weed species, species not specified	Multispectral, RGB-D	Classification, detection and segmentation	Chebrolu et al. (2017)
Food crops and weeds	Common beet, Carrot, Zucchini, Pumpkin, Radish and Black radish	*Chenopodium album L.* *Galium aparine* *Thlaspi arvense* *Capsella bursa-pastoris* *Matricaria perforata* *Polygonum convolvulus* *Viola arvensis* *Galinsoga parviflora*	RGB	Detection	Sudars et al. (2020)
Open plant phenotype database	Not specified	47 weed species. Names given in the paper	RGB	Detection	Madsen et al. (2020)
Sugar beet and hedge bindweed	Sugar beet	*Convolvulus sepium*	RGB	Detection	Gao et al. (2020)
Crop/Weed	Carrot	Weeds. Names do not specified	Multispectral	Segmentation	Haug and Ostermann (2015)
Carrot/Weed	Carrot	Not specified	RGB	Segmentation	Lameski et al. (2017)
GrassClover	Red clover and white clover	*Lolium perenne* *Taraxacum officinale* *Capsella bursa-pastoris* *Cirsium arvense*	RGB	Segmentation	Skovsen et al. (2019)
WeedNet	Crop. Name do not specified	Weeds. Names do not specified	Multispectral	Segmentation	Sa et al. (2018)
Rice seedling and weed	Rice	*Sagittaria trifolia*	RGB	Segmentation	Ma et al. (2019)
Crop and weeds	Corn and common bean	*Chenopodium album* *Matricaria chamomilla* *Brassica nigra* *Lolium perenne*	RGB	Segmentation	Champ et al. (2020)
Sunflower dataset	Sunflower	Not specified	Multispectral	Segmentation	Fawakherji et al. (2021)

Data preparation could also include transforming the images from RGB to a different color space or color index images for training models. Wang et al. (2020) segment crop and weed plants training DL architectures using images in the *YCrCb* and *YCgCb* color space and from eight distinct color indices. Also, Milioto et al. (2018) used 14 distinct image representations, including color spaces and color indices for weed and crop segmentation. On the other hand, image resizing is done as dataset preparation when scholars pretend to make public their data. As a reference, the datasets provided in Table 1.4 have been configured with fixed image size. However, we argue that datasets whose size of their images was manipulated have lost prominent features, preventing the users from exploring with other techniques.

Training stage

The training stage encloses the steps of image pre-processing, training DL architecture, and testing of the model. Image resizing is often the first step in image pre-processing if the data has not been resized previously. The size of the images depends on the requirements of the model architecture. Common image sizes, in pixels, found for weed recognition are 64×64 (Milioto et al., 2018; Reedha et al., 2022), 128×128 (Dyrmann et al., 2016; Espejo-Garcia et al., 2020), 224×224 (Jiang et al., 2020; Olsen et al., 2019; Wang et al., 2023), 256×256 (dos Santos Ferreira et al., 2017; Tang et al., 2016; Zou et al., 2022) and 512×512 (Janneh et al., 2023; Zhang et al., 2023). The size of images also contributes to the performance of the networks. Sahin et al. (2023) found that the *Sunflower* dataset (Fawakherji et al., 2021) was better segmented using 704×704 image size than a size of 512×512 through the UNet architecture. Other works have found the better performance of DL models as image resolution was increased (Sabottke and Spiele, 2020; Thambawita et al., 2021). To avoid the loss of prominent features during the training of DL architectures, some scholars have opted to use the original size of the images and work with patches. Fawakherji et al. (2020) patchify the dataset *Sunflower* (Fawakherji et al., 2021) and *Sugar beet* (Chebrolu et al., 2017) for segmentation of crop and weeds.

Although DL architectures have the potential to learn multiple features, some scholars argue that eliminating the background (soil, rocks, or human body parts) contributes to increasing the performance of the models on specific tasks because the background features do not interfere with the features of the plants (KC et al., 2021). Commonly, if background removal is applied over images, it comes along with image enhancement and/or image denoising actions to obtain a foreground in most corresponding to alive vegetation. Le et al. (2020a) implemented $E \times G - E \times R$ index to keep solely green vegetation pixels on images of wild radish and barley. They also implemented opening and closing morphological operations (Soille,

2004) to remove noise in the images. The NDVI also was used for background removal and opening and closing morphological operations for noise elimination Milioto et al. (2017) over the *sugar beet* dataset (Chebrolu et al., 2017) to detect crops/weeds.

Nonetheless, image enhancement and denoising have also been employed independently as pre-processing techniques during the training stage for weed/crop recognition. Lottes et al. (2020) removed the noise of input images by applying a Gaussian blur kernel and enhanced them by standardizing the channels by subtracting the mean of channel values and dividing by standards deviation of the channel values in order to minimize the influence of changes of the environment for crop/weed classification. Furthermore, the effects of image enhancement and denoising have been evaluated on the performance of CNNs. Lottes et al. (2018) found that applying a Gausian kernel to remove noise, standardizing and normalizing the pixels at zero-center values helped the networks to increase the generalization capabilities. Nkemelu and amd Nancy Lubalo (2018) also reported that applying a Gaussian kernel for smoothing the training images and removing the background, the classification of seedling weeds through a CNN increased around 12.40 %. Nonetheless, all the actions carried out in the image pre-processing step into the training stage increase the training time of the networks.

After having chosen or designed a DL architecture for crop/weed identification, the training process involves the “syntonization” of the weights, parameters, and hyper-parameters of the networks over a portion of the experimental dataset (training and validation). Afterward, the models should be tested over another portion of the dataset. In the literature, for weed identification, it is common to find splitting rates of datasets of 7:2:1 (Wang et al., 2023; Zhang et al., 2023), 6:2:2 (Jiang et al., 2023; Vaidhehi and Malathy, 2022; Zou et al., 2022), 8:1:1 (Picon et al., 2022; Subeesh et al., 2022) for training, validation and testing, respectively.

1.1.3.1. Classification, detection and segmentation of crops and weeds

The design of the DL networks is done according to the recognition approach wanted, such as classification, detection, and segmentation. Because in recent years, plenty of works have been reported for weed recognition based on deep learning, below are listed some works that cover weed recognition solely in natural conditions, in which the networks used, the crop/weed species, and the corresponding descriptive metrics are highlighted.

Classification

Classification is the task of assigning a class to a single object present within an image (Skansi, 2018). This means that classification networks do not provide the spatial localization of the crop and weed plants in the field. Since the AlexNet was reported in 2012, modern classification architectures are composed of convolutional layers for feature extraction and fully connected layers for classification of the features (Santosh et al., 2022). Since that year, CNN architectures have suffered diverse modifications, such as the network depth, which has been achieved by reorganizing the processing units, from plain stacking of convolutions to the development of new blocks like inception, residual connections, and dense blocks. On the other hand, transformer architectures for crop/weed recognition, which practically arose in 2017, have also evolved. Therefore, works settled in Table 1.5 for weed recognition in open fields include the following architectures: AlexNet and VGGNet, which have been designed stacking the convolution operations; and the modern Inception-based CNNs, ResNet and its variants, and finally DenseNet; whereas transformers include Vision Transformer (ViT) and Swin Transformer, principally.

Detection

Detection consists of localizing the instances of an object in a given image and designating each object a class from a group of predefined classes (Amjoud and Amrouch, 2023). DL architectures specialized in detection are integrated with a backbone, which is a CNN designed to extract relevant features of the input images; a localization algorithm, which provides the spatial localization of the instances in an image; and finally, a fully connected networks, which classify the region proposal that comes from the localization algorithm. The main difference among detection architectures is principally its localization algorithms.

Table 1.5: Works over the classification of crops/weeds in natural fields based on CNNs and transformers.

Crop	Weed species	Model	Accuracy	Reference
Soybean	Grass and broadleaf weeds	AlexNet	99.5 %	dos Santos Ferreira et al. (2017)
Sugar beet	Volunteer potato	AlexNet VGG19 GoogleNet ResNet50 ResNet101 InceptionV3	97.9 % 98.7 % 97.3 % 97.2 % 98.5 % 94.8 %	Suh et al. (2018)
Corn	*Cirsium setosum,* *Chenopodium album,* *Cyperus esculentus* *and Poa annua*	AlexNet VGG16 ResNet101	93.0 % 95.8 % 96.5$	Jiang et al. (2020)
Lettuce	*Cirsium setosum,* *Chenopodium album,* *Cyperus esculentus,* *and Poa annua*	AlexNet VGG16 ResNet101	96.65 % 97.61 % 98.25 %	Jiang et al. (2020)
Corn, potato and sunflower	*Alopecurus myosuroides Huds,* *Amaranthus retroflexus L.,* *Avena fatua L.,* *Chenopodium album L.,* *Lamium purpureum L.,* *Matricaria chamomila L.,* *Setaria spp.,* *Solanum nigrum L.* and *Stellaria media Vill.*	VGG16 ResNet50 Xception	82.0 % 97.0 % 98.0 %	Peteinatos et al. (2020)
Canola	Wild radish	VGG16 VGG19 ResNet50 InceptionV3	91.55 % 989.55 % 89.73 % 90.87 %	Le et al. (2020a)
Corn and soybean	*Xanthium strumarium,* *Setaria viridis,* *Amaranthus retroflexus* *and Ambrosia trifida*	VGG16 ResNet50 InceptionV3	99.90 % 97.80 % 96.70 %	Ahmad et al. (2021)
Bell paper	Multiple weeds. Species are not specified	AlexNet GoogleNet InceptionV3 Xception	96.7 % 95.9 % 97.7 % 96.8 %	Subeesh et al. (2022)
Beet, Parsley and Spinach	Multiple weeds. Species are not specified	Visual Transformer B-16 EfficientNet B0 ResNet50	99.28 % 96.53 % 97.54 %	Reedha et al. (2022)
Corn seedling	*Cyperus rotundus L.,* *Amaranthus retroflexus L.,* *Abutilon theophrasti Medicus,* *Portulaca oleracea L.,* *Chenopodium album L.,* *Cirsium setosum* and *Descurainia sophia L.*	VGG16 ResNet50 DenseNet121 SE-ResNet50 EfficientNetV2 Swin Trasnformer	96.12 % 96.33 % 96.73 % 96.94 % 97.35 % 97.96 %	Wang et al. (2023)
Corn	Multiple weeds. Species are not specified	MobileNetV2 InceptionV3	90.0 % 71.0 %	Wessner et al. (2023)

In Table 1.6 detection specialized architectures that have been implemented over crops and weeds are listed. The most often used architectures for this task in natural fields are DetectNet, YOLO in its different variants, SSD, Fast Region-Based Convolutional Neural Network (R-CNN), Faster Region-based Convolutional Neural Network (Faster R-CNN), and RetinaNet.

Segmentation

Segmentation is divided into semantic segmentation and instance segmentation. Semantic segmentation is the task of assigning a label to each pixel (classification) of an image from predefined classes grouping the instances into classes, whereas instance segmentation not only assigns a label to each pixel of an image from predefined classes but also distinguishes between different instances of the same class (Michelicci, 2019).

The DL architectures for semantic segmentation are grouped into Region Proposal, Fully Convolutional Neural Networks (FCNN), and transformer-based. Section 3.1.3 will cover some of these approaches. Region proposal-based architectures extract multiple region proposals from an input image, and then the pixels of the proposal are labeled with the highest score label the proposal contains. The most well-known architectures in this group are Region-based Convolutional Neural Network (R-CNN), Fast R-CNN, Faster R-CNN, and Mask Region-based Convolutional Neural Network (Mask R-CNN). On the other hand, FCN-based segmentation is divided as follows, encoder-decoder networks, such as UNet and SegNet; Networks with Dilated Autrous Convolutions, like DeepLab networks; Feature fusion networks, such as ParseNet; Multi-Scale Feature and Pyramid Architectures, like PSPNet and CNet (Soylu et al., 2023). Since 2023, Transformers have been used for the segmentation of crops and weeds. Those reported in the literature are Vision Transformer, Swin Transformer, SegFormer, Segmenter, and Swin-UNet. All these networks follow an encoder-decoder structure. Table 1.7 provides some DL architectures used for the segmentation of crops and weeds in outdoor conditions.

It is worth mentioning that from the works reported in Table 1.7 for segmentation of corn plants, Picon et al. (2022) worked under high density of plants, whereas the density of plants in the work of Zhang et al. (2023) was lower. The crop and weed plants were also at the seedling stage in this last work. Therefore, the magnitude difference of the metric mIoU among these works is attributable to these factors.

Table 1.6: Works over the detection of crops/weeds in natural fields based on CNNs.

Crop	**Weed species**	**Model**	**IoU**	mAP_{50}	**Reference**
Wheat	Multiple weeds. Species are not specified	DetectNet	64.0 %	–	Dyrmann et al. (2017)
Sugar beet	*Convolvulus sepium*	YOLO 3	–	83.20 %	Gao et al. (2020)
Corn seedling	Weeds are not included	YOLO 3-tiny 3	84.0 %	–	Liu et al. (2020)
		YOLO 3	76.0 %	–	
Corn and soybean	*Xanthium strumarium, Setaria viridis, Amaranthus retroflexus and Ambrosia trifida*	YOLO 3	–	54.3 %	Ahmad et al. (2021)
Lettuce	*Sonchus brachyotus, Plantago asiatica L., Malachium aquaticum L. Avena fatua and Veronica officinalis*	SE-YOLO 5	–	97.1 %	Zhang et al. (2022)
		YOLO 5	–	96.2 %	
		SSD-VGG16	–	86.2 %	
		SSD-MobileNetV2	–	95.1 %	
		Faster R-CNN-ResNet50	–	81.5 %	
		Faster R-CNN-VGG16	–	83.8 %	
Tomato	*Cyperus rotundus L., Echinochloa crus galli L., Setaria verticillata L., Portulaca oleracea L.* and *Solanum nigrum L.*	RetinaNet	–	92.75 %	López-Correa et al. (2022)
		YOLO 7	–	83.08 %	
		Faster R-CNN	–	92.13 %	
Sesame	Multiple weeds. Species are not specified	Fast R-CNN	–	72.96 %	Chen et al. (2022)
		SSD	–	78.83 %	
		EfficientDet-d0	–	80.64 %	
		YOLO 3	–	65.81 %	
		YOLO 4	–	91.19 %	
		YOLO 4-tiny	–	81.71 %	
		YOLO-sesame	–	96.16 %	
Cotton, soybean and corn. (Crops do not detected)	*Ipomoea purpurea Ipomoea hederacea Urochloa texana Sorghum halepense Amaranthus palmeri Euphorbia humistrata Panicum fasiculatum*	YOLO 4	–	65.83 %	Sapkota et al. (2022)
		Faster R-CNN	–	59.33 %	

IoU-intersection over union; mAP_{50}- mean average precision considering an $IoU = 0.5$.

1.2. Justification

Agricultural practices for corn crops should be improved now to ensure food security in the future. Weeds significantly reduce the harvest volume of this ce-

Table 1.7: Works over the segmentation of crops/weeds in natural fields based on CNNs and transformers.

Crop	Weed species	Model	mIoU	DSC	Reference
Rice seedling	*Sagittaria trifolia*	SegNet-VGG16	91.80 %	–	Ma et al. (2019)
		FCN	53.80 %	–	
		UNet	53.00 %	–	
Rice seedling	*Sagittaria trifolia*	UNet	59.67 %	74.74 %	Khan et al. (2020)
		SegNet	67.41 %	80.53 %	
		FCN-8s	54.78 %	70.78 %	
		DeepLabV3	67.60 %	80.67 %	
		CED-Net	71.05 %	83.08 %	
Canola	Broadleaf weeds and narrowleaf weeds. Species are not specified	UNet-VGG16	78.05 %	99.52 %	Asad and Bais (2020)
		UNet-ResNet50	82.74 %	99.64 %	
		SegNet-VGG16	79.20 %	99.55 %	
		SegNet-VGG16	82.88 %	99.29 %	
Rice	Broadleaf weeds and narrowleaf weeds. Species are not specified	PSPNet-ResNet50	62.44 %	–	Kamath et al. (2022)
		UNet-ResNet50	51.35 %	–	
		SegNet-VGG16	31.88 %	–	
Wheat	Not included	DeepLabV3+ - ResNet50	77.50 %	86.30 %	Zenkl et al. (2022)
Corn	*Setaria verticillata*, *Digitaria sanguinalis*, *Echinochloa crus-galli*, *Abutilon theophrasti*, *Chenopodium albums* and *Amaranthus retroflexus*	PSPNet-ResNet50	–	45.33 %	Picon et al. (2022)
		Dual PSPNet-ReSNet50	–	47.97 %	
Wheat	*Trigonotis peduncularis*, *Rorippa indica (L.) Hiern*, *Cirsium setosum* and *Chenopodium album L.*,	Modified UNet	92.84 %	–	Zou et al. (2022)
		UNet	92.45 %	–	
		SegNet	78.68 %	–	
		FCN	72.15 %	–	
Canola	Broadleaf weeds and narrowleaf weeds. Species are not specified.	UNet	60.95 %	–	Das and Bais (2021)
		UNet-ResNet50	63.34 %	–	
		SegNet	65.21 %	–	
		SegNet-ResNet50	59.37 %	–	
		DeepLabV3+	61.61 %	–	
		DeepLabV3+ - VGG19	56.47 %	–	
		DeepVeg	76.79 %	–	
Soybean	Broadleaf weeds and narrowleaf weeds. Species are not specified.	DeepLabV3+	92.80 %	–	Xu et al. (2023)
		DeepLabV3	96.70 %	–	
		FCN	91.90 %	–	
		UNet	92.20 %	–	
		FastFCN	93.10 %	–	
		Vision Transformer	93.20 %	–	
		Swin Transformer	93.00 %	–	
		ResNet101-DSASPP	93.90 %	–	

mIoU- mean intersection over union; DSC - dice similarity coefficient.

Table 1.7: Continue.

Crop	Weed species	Model	mIoU	DSC	Reference
Grass	*Trifolium repens*, *Ambrosia artemisiifolia*, *Digitaria*, *Taraxacum*, *Glechoma hederacea*, *Chenopodium album*, *Amaranthus*, *Plantago asiatica L.*, *Festuca arundinacea* and *Unknown weeds*	Swin Transformer	65.41 %	–	Jiang et al. (2023)
		SegFormer	65.74 %	–	
		Segmenter	59.24 %	–	
Corn	Broadleaf weeds and narrowleaf weeds. Species are not specified.	DeepLabV3+	90.48 %	–	Zhang et al. (2023)
		PSANet	91.67 %	–	
		Mask R-CNN	91.97 %	–	
		Swin-UNet	92.03 %	–	
		Improved Swin-UNet	92.75 %	–	

mIoU- mean intersection over union; DSC - dice similarity coefficient.

real, and they are most often controlled by spraying herbicides, which pollute the environment. SSWM is an alternative for mitigating this pollution. Nonetheless, localizing weed plants in open corn fields is still a challenge. In this way, the nowadays reported studies have relied on supervised vision ML to tackle this trouble. Most works have trained the models with datasets acquired under controlled light conditions and low background variability, meaning that soil appearance and straws do not change, or even datasets with scarce plant species are used. To implement DL-based vision systems in natural field conditions, a considerable quantity of images captured at different scenarios and growing stages of the plants are needed so that the performance of the systems does not decay with new data. It was found in the literature that DL algorithms were trained to classify individual plant species. Also, vision systems for detecting weeds and crops have been implemented in crops different than corn, and a low number of works have been carried out on corn, which is of economic and sociocultural importance. Therefore, in this work, classification and detection algorithms based on segmentation networks have been implemented for detecting corn plants (Crop), common narrow-leaf weeds (NLW), and broad-leaf weeds (BLW) from multi-plant images. Moreover, to train the proposed DL models, a large dataset of images acquired in a typical cornfield under natural environmental conditions has been created. This dataset contains nine plant species grouped into three classes.

1.3. Objectives

1.3.1. General objective

Developing a vision system that uses DL architecture to detect weeds in natural corn fields and implementing it on a mechatronic platform for real-time weed control.

1.3.2. Specific objectives

- To generate an image dataset of common weeds that grow in corn fields.
- To explore shallow and deep learning algorithms to extract features of weed plants for detecting them in the field.
- To explore deep learning algorithms for semantic segmentation of weeds and classic algorithms for object location.

1.4. Hyphotesis

Through a DL-based vision system, it is possible to detect in real-time (20 fps) at least 95 % of the weed plants situated in the crop fields of Aguascalientes, Mexico.

Chapter 2

Theoretical framework

This section provides an in-depth exploration of the theoretical underpinnings that serve as the framework for this research. It provides a comprehensive understanding of the technical knowledge involved in the algorithms used to classify and segment corn and weed plants.

2.1. Artificial Intelligence

Artificial Intelligence (AI) is a branch of computer science that aims to create intelligent machines capable of mimicking human cognitive functions, including learning, problem-solving, and decision-making (Alzubaidi et al., 2021). The development of AI involves the creation of algorithms and models that enable machines to process information, recognize patterns, and adapt their behavior accordingly. The importance of AI lies in its transformative potential across various industries, offering unprecedented opportunities for innovation and efficiency. AI technologies can enhance decision-making processes, automate repetitive tasks, and analyze vast datasets at speeds beyond human capability. From healthcare and finance to manufacturing and entertainment, the applications of AI are diverse and expansive, spanning predictive analytics, natural language processing, image recognition, and autonomous systems.

AI, ML, DL, and ANN are interrelated technologies often used interchangeably, leading to confusion about their differences. It is, therefore, valuable to clarify that deep learning is a specialized subset of ML, which in turn is a subset

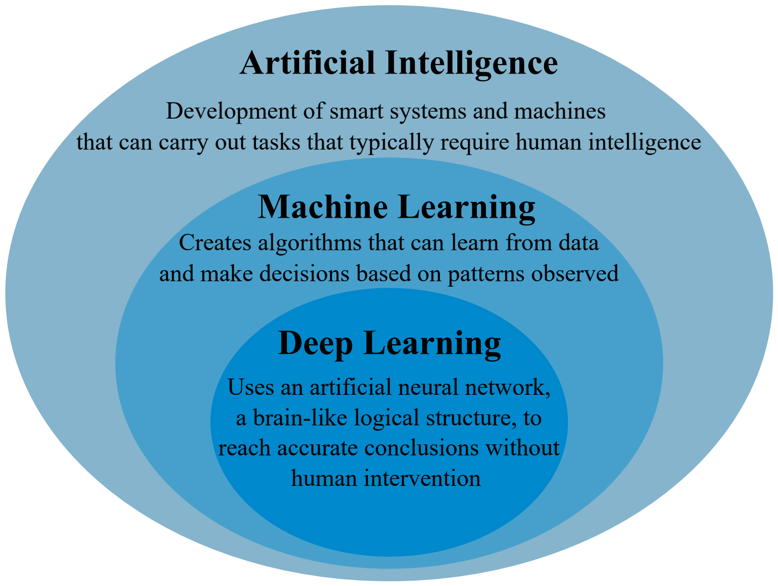

Artificial Intelligence
Development of smart systems and machines that can carry out tasks that typically require human intelligence
Machine Learning
Creates algorithms that can learn from data and make decisions based on patterns observed
Deep Learning
Uses an artificial neural network, a brain-like logical structure, to reach accurate conclusions without human intervention

Figure 2.1: The artificial intelligence family. Machine learning is a subset of artificial intelligence and deep learning is a subset of machine learning.

of AI, as shown in Figure 2.1. ML needs human intervention for feature extraction steps and commonly works with small datasets. In case the number of features increases, the performance of ML models usually decreases. In contrast, deep learning does not need human intervention because the models extract and learn features in one step. However, they need large datasets to obtain powerful models (Mueller and Massaron, 2022; Taulli, 2019)

2.2. Machine learning

To implement ML-based object recognition, it is necessary to first extract "features" from the objects in question (weeds and crops for this work), i.e., $\varphi : \mathcal{T} \to \mathcal{X}$ where φ represents the mapping from "object" to features, $\mathcal{T}$ is the set of objects, and $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ is the set of "N" feature vectors corresponding to each object. It is worth mentioning that each $\boldsymbol{x} \in \mathcal{X}$ takes the form of d-dimensional vectors, i.e., $\boldsymbol{x} \in \mathbb{R}^d$, where $d \in \mathbb{N} \setminus \{0\}$ is the dimension size, then, $\mathcal{X} \in \mathbb{R}^{N \times d}$.

Once having a prepared dataset, the following step under the ML approach is to find out a model with a prediction function $f(\cdot)$, based on some measure of "quality", that maps $\mathcal{X}$ to the output $\mathcal{Y}$, namely, $f : \mathcal{X} \times \Theta \rightarrow \mathcal{Y}$, where Θ represents model's parameters. This is expressed as follows,

$$\mathcal{Y} = f(\mathcal{X}, \Theta) \tag{2.1}$$

Mainly, in this thesis, the classification of crop and weed plants was first performed using a ML approach, specifically, a *supervised learning* method. Supervised learning is a ML paradigm where a model is trained on a labeled dataset, meaning that the algorithm learns from input-output pairs provided during training. Therefore, the output was a set of labels, i.e., $\mathcal{Y} = \{y_n\}_{n=1}^{N}$, in which each label $y_n \in \mathbb{R}$ is associated with each feature vector $\boldsymbol{x}_n \in \mathbb{R}^d$. Hence, we have proposed a model that implements the prediction function $f : \mathcal{X} \times \Theta \rightarrow \mathcal{Y}$. Then, the Equation 2.1 can be expressed as:

$$y_n = f(\boldsymbol{x}_n, \Theta), \quad \forall n \in \{1, ..., N\} \tag{2.2}$$

Common shallow learning approaches in supervised learning include classic algorithms such as linear regression, decision trees, SVM, k-nearest neighbors (KNN), random forest (RF), and Bayesian classifiers (Bishop, 2006).

For a typical image classification problem, the output is a set of L labels known as classes, $\mathcal{Y} = \{1, 2, 3, ..., L\}$. The input $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$, could be the set of vectors with the pixel intensity of images. In this way, having color images with three channels ($c = 3$) of size $w \times h$ pixels. Then, $\mathcal{X} \in \mathbb{R}^{N \times w \times h \times c}$ belongs to a very high-dimensional space. Thus, learning a prediction function ($f : \mathcal{X} \times \Theta \rightarrow \mathcal{Y}$) for mapping the input images to labels is relatively challenging.

Remarkably, this thesis adopted the well-known texture feature operator $LBP_{P,R}^{riu2}$ (Ojala et al., 2002). This decision allows the input to belong to a lower-dimensional space. Then, utilizing an SVM for classification tasks becomes a feasible alternative.

2.2.1. Local binary pattern

LBP was introduced by Ojala (Ojala and Pietikainen, 1999; Ojala et al., 1996). This descriptor specializes in texture analysis of gray-scale images. What distinguishes LBP is its monotonic gray-scale transformation and its illumination and rotation invariance (Hamouchene et al., 2014).

The LBP operator works by comparing the value of a center pixel with those of its surrounding pixels. If the center pixel value is greater than the surrounding pixel value, it is assigned a value of 0; otherwise, it is assigned a value of 1. The LBP operator is defined as follows,

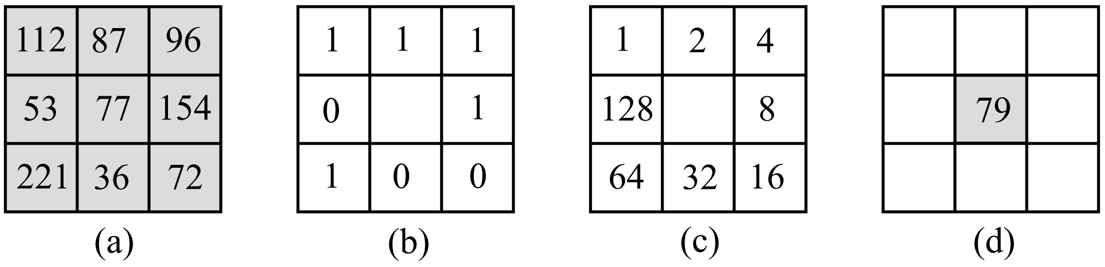
$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{2.3}$$

where g_c represents the gray value of the center pixel, g_p is the gray value of the neighbors, P is the number of pixels in the circular neighborhood of radius R, and $s : \mathbb{Z} \to [0, 1]$ is a function defined as,

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}, \quad x \in \mathbb{Z} \tag{2.4}$$

Figure 2.2 calculates the LBP "code" of a 3×3 gray-scale image window. First, the intensity of the center pixel ($g_c = 77$) is compared with the intensity of each of the eight surrounding pixels (g_p). The pixel intensities are shown in Figure 2.2(a). Starting from the top left corner of the window, when the difference value of ($g_p - g_c$) is greater than 0, it is considered 1; otherwise, it is considered 0. From this process, an 8-bit binary pattern obtained is 11110010, as shown in Figure 2.2(b). The weights of Figure 2.2(c) are calculated by the operation 2^p, a factor of Equation (2.3). Then, the binary pattern (Figure 2.2(b)) is element-wise multiplied with the weights (Figure 2.2(c)), and the products are summed to obtain an LBP code, which in this case is 79. Finally, this LBP code is replaced by the central pixel of the window (Figure 2.2(d)). Figure 2.3 shows an instance RGB image whose pixels were labeled with LBP values.

The LBP algorithm above reflects the texture features of an image utilizing the histogram of the LBP codes. This histogram is enclosed in a vector of 256



112	87	96
53	77	154
221	36	72

(a)

1	1	1
0		1
1	0	0

(b)

1	2	4
128		8
64	32	16

(c)

	79	

(d)

Figure 2.2: Example for computing the LBP code. (a) Fraction of gray-scale image. The numbers represent the intensity of the pixels. (b) 8-bit binary pattern. It is computed comparing the intensity of the center pixel (g_c) with the intensity of the eight surrounding pixels (g_p), starting from the top left corner, using Equation 2.4. (c) Weighs for computing patterns, computed by the operation 2^p, where $p = 0, ..., 7$, and (d) the LBP code of the central pixel calculated through Equation 2.3.

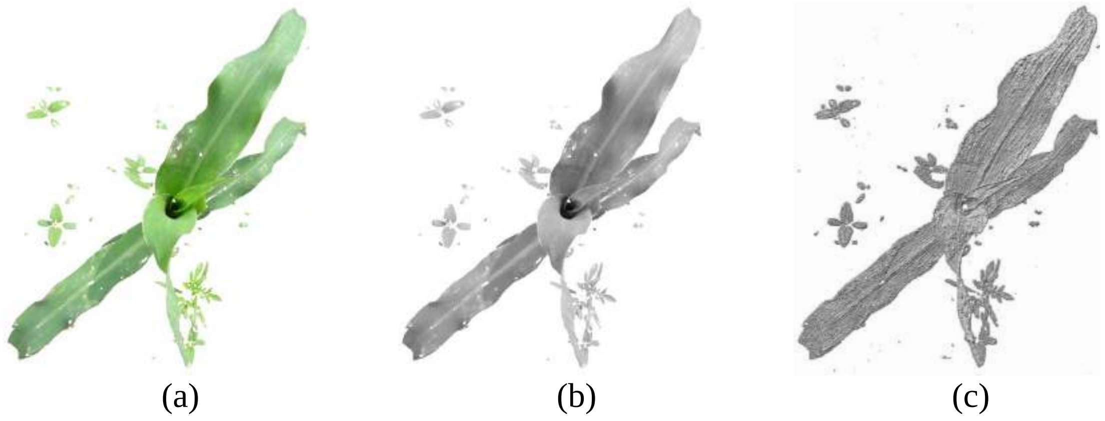


Figure 2.3: Visualization of an RGB image whose pixels have been labeled with LBP codes. (a) Input RGB image. (b) Gray color space. (c) Labeled image with LBP codes.

possible patterns ($\boldsymbol{x}_n \in \mathbb{R}^{256}$). To further analysis, the formed set $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ is usually normalized for subsequent training of the selected prediction function.

The original LBP operator fails to capture other outstanding features because only a 3×3 neighborhood is considered and it always contemplates the same number of surrounding pixels, which is a drawback. Additionally, not all 256 possible patterns are necessary to extract the most important features (Hamouchene et al., 2014; Le et al., 2019). Ojala et al. (2002) improved the original LBP algorithm by considering exclusively the number of transitions between 0 and 1 or 1 and 0, denoted by the "uniform" measure U. This new algorithm has been named rotation-invariant uniform local binary pattern ($LBP_{P,R}^{riu2}$).

The following Figure 2.4 clarifies $LBP_{P,R}^{riu2}$, in which the black circles represent a bit value of 1 and the white circles represent a bit value of 0 in the 8-bit output of the traditional LBP. Then, this algorithm adopts the uniform patterns with zero ($U = 0$) or two transitions ($U = 2$). When the pattern has zero transition, it is a compound of either ones or zeros, such as 11111111 (Figure 2.4(a)) and 00000000 (Figure 2.4(b)), respectively. A pattern with two transitions transits from 0 to 1 or from 1 to 0, such as 00111100 (Figure 2.4(c)). Non-uniform patterns have more than two transitions, such as 10110111, represented in Figure 2.4(d). In this way, the $LBP_{P,R}^{riu2}$ descriptor is denoted as follows,

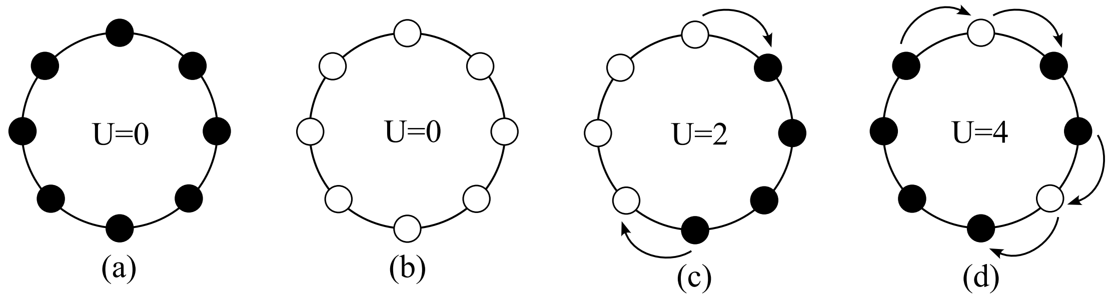


Figure 2.4: Rotation invariant binary pattern instances in a circular neighbor set of eight pixels. The black circles represent a bit value of 1 and the white circles represent a bit value of 0. (a) Uniform pattern with cero transitions (11111111), (b) uniform pattern with cero transitions (00000000), (c) uniform pattern with two transitions (00111100), and (d) nonuniform pattern. This type of pattern have more than two transitions from 0 to 1 or from 1 to 0.

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c), & \text{if } U(LBP_{P,R}) \leq 2 \\ P+1, & \text{otherwise} \end{cases} \tag{2.5}$$

where,

$$U(LBP_{P,R}) = \left| s(g_{P-1} - g_c) - s(g_0 - g_c) \right| + \sum_{p=1}^{P-1} \left| s(g_p - g_c) - s(g_{p-1} - g_c) \right| \tag{2.6}$$

$LBP_{P,R}^{riu2}$ breeds individual datapoints $\boldsymbol{x}_n \in \mathbb{R}^{P+2}$, that reduce the feature space and increase the speed of LBP. For instance, if a window of eight circular neighbor pixels is used, then $\boldsymbol{x}_n \in \mathbb{R}^{10}$.

2.2.2. Support vector machines

SVM, a supervised ML algorithm, solves the two-classes classification problem using the following linear model,

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{2.7}$$

where the parameters $\mathbf{w}$ and b, the weights and bias, respectively, are calculated from a training dataset of input vectors $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^N$ with corresponding target values $\mathcal{Y} = \{y_n\}_{n=1}^N$, where $y_n \in \{-1, 1\}$, in such a way that new data points x_n are classified according to the "sign" of $f(\mathbf{x})$. The SVM approaches the classification problem by maximizing the margin distance, defined as the distance between the decision boundary and the closest samples, as shown in Figure 2.5.

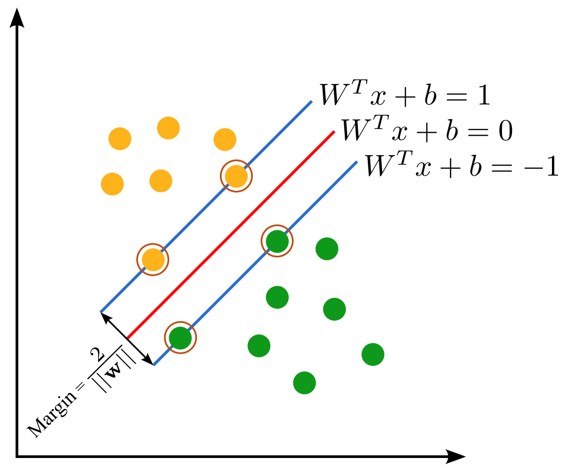


Figure 2.5: Illustration of the decision boundary of SVM for two classes.

The margin is calculated by an optimization process of the parameters $\mathbf{w}$ and b as follows:

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \left\{ \frac{1}{||\mathbf{w}||} \min_n \left[y_n(\mathbf{w}^T\mathbf{x} + b) \right] \right\} \tag{2.8}$$

To solve this optimization problem, a Lagrange multiplier is needed,

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n\{y_n(\mathbf{w}^T\mathbf{x} + b) - 1\} \tag{2.9}$$

where $\mathbf{a}$ is a vector of multipliers, whose elements $a_n \geq 0$, and N are the input vectors. To simplify Equation (2.9), the derivatives with respect to $\mathbf{w}$ and b are computed. Next, these derivatives are set equal to zero, resulting,

$$\mathbf{w} = \sum_{n=1}^{N} a_n y_n \mathbf{x} \tag{2.10}$$

$$0 = \sum_{n=1}^{N} a_n y_n. \tag{2.11}$$

Thus, using these conditions, Equation (2.9) can be expressed as follows,

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m y_n y_m K(\mathbf{x}_n, \mathbf{x}_m) \tag{2.12}$$

with constraints,

$$a_n \geq 0, \quad n = 1, \ldots, N, \tag{2.13}$$

$$\sum_{n=1}^{N} a_n y_n = 0 \tag{2.14}$$

where K is a kernel function, which transforms a non-linearly separable space to a linear separable one, and a_n is a constant known as the Lagrange multiplier.

2.3. Deep learning

DL encloses models with low training parameters, such as shallow ANNs, that usually comprehend two or three layers. On the contrary, deep learning encompasses a large family of models that contain more complex functions. For instance, if a shallow ANN is added with more hidden layers, deep learning trait these problems. Therefore, the working principle of shallow and deep ANNs is first provided to further understand CNNs and transformers.

2.3.1. Artificial Neural Networks

The simplest representation of an ANN is the perceptron, which was proposed by Rosenblatt (1958). The perceptron is constituted by a single artificial neuron in one layer, as shown in Figure 2.6.

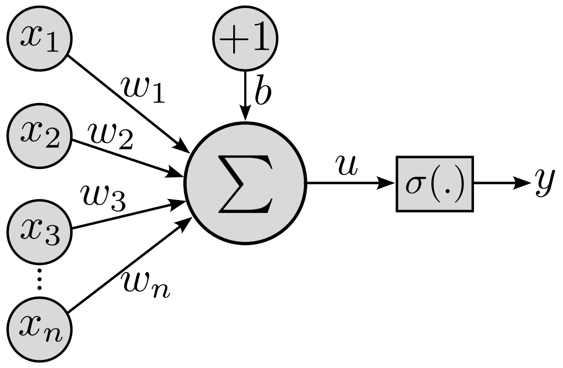


Figure 2.6: The perceptron network representation. The inputs (x_i) are pondered by their corresponding weights (w_i) to regulate how much of the initial value will be forwarded to a given neuron. The resulting value is added with the parameter *bias* (b) for obtaining an activation potential (u) that is then an argument of the activation function $\sigma(\cdot)$. The final value y is produced by a neuron given a set of input signals.

The basic elements elements of an ANN are the following:

- The input signals $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$, which are the magnitude of the variables that describe a particular application.
- The weights $\mathcal{W} = \{\boldsymbol{w}_n\}_{n=1}^{N}$ are the numbers used to regulate how much of the initial value will be forwarded to a given neuron.
- Linear aggregator ($\sum$) congregates all input signals pondered by the weights.
- The bias (b) is a parameter employed to define the requisite threshold that the outcome from the linear aggregator must possess to initiate a triggering response in relation to the neuron output.
- Activation function ($\sigma(\cdot)$) is used to incorporate non-linearity into the network by limiting the neuron output within a reasonable range of values.

- Output signal (y) is the final value produced by a neuron given a set of input signals.

McCulloch and Pitts (1988) proposed the following expressions that describe the mathematical notation performed by the perceptron:

$$u = \sum_{i=1}^{n} w_i \cdot x_i + b \tag{2.15}$$

$$y = g(u) \tag{2.16}$$

where x_i is an input to the network, w_i is the weight related with the i-th input, b is the bias, $g(\cdot)$ is the activation function and u is the activation potential.

The working principle of a single neuron starts when the inputs (x_i) are pondered by their corresponding weights (w_i) to regulate how much of the initial value will be forwarded to a given neuron. The resulting value is added with the parameter *bias* (b) for obtaining an activation potential (u) that is then an argument of the activation function ($\sigma(\cdot)$). The information in the perceptron network always flows from the input layer to the output layer, at this stage, do not exist feedback from the output neuron to the input.

2.3.1.1. Multilayer Perceptron Network

An MLP, a *feedforward neural network*, arranges multiple neurons in layers. These layers are usually known as the input , hidden and output layer; in which every layer is represented by nodes connected to all nodes to the next layer, as shown in Figure 2.7. The weight parameters are indicated as links between the nodes. The *bias*(b) is also represented by links coming as inputs to every node of the hidden and output layers. In this network, the information flows from the input layer to the output layer during forward propagation, which is represented by the arrows.

For instance, for constructing a three-layer network, M linear combination of the input variables ($x_1, x_2, ...x_D$) is constructed first, in the form,

$$u_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)} \tag{2.17}$$

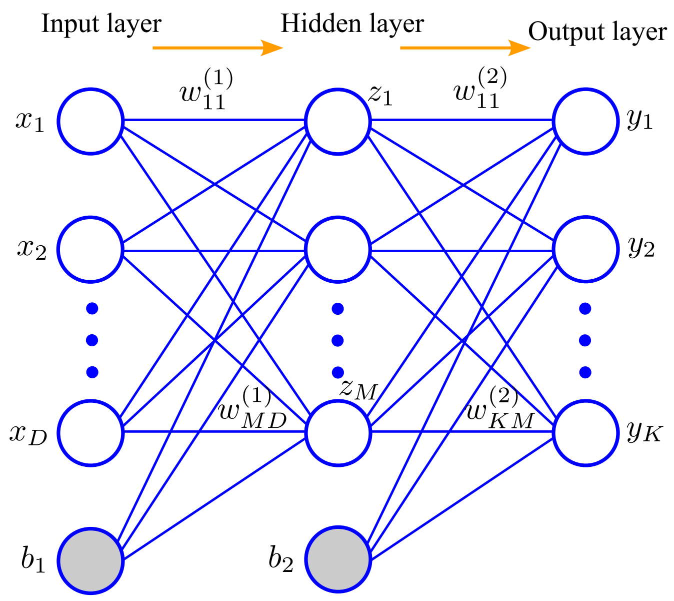


Figure 2.7: A simple neural network representation of three layers.

where $j = 1, ..., M$ and the superscript (1) indicates that these parameters corrrespond to the first layer of the network. The parameters w_{ji} and b are the weights and the bias, correspondingly. Every merging activation potential u is then transformed using any differentiable, non-linear activation function $\sigma(\cdot)$ to obtain,

$$z_j = \sigma(u_j) \tag{2.18}$$

where z_j is the output of the hidden units. It is worth mentioning that the most often used non-linear activation function $\sigma(\cdot)$ in the hidden layers of an ANN is the logistic sigmoid. However, different layers may have different activation functions. Nonetheless, all neurons in the same hidden layer shall have the same activation function. The next section covers different activation functions that are commonly used in an ANN for the hidden layers and output layer.

Continuing, each z_j is anew linearly combined to obtain output activation for the second layer,

$$u_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + b_k^{(2)} \tag{2.19}$$

where $k = 1, ..., K$, and K is the total number of outputs. Then, the output activation potentials u_k are transformed using an appropriate activation function to obtain the output of the network y_k.

$$y_k = \sigma(u_k) \tag{2.20}$$

In this case, for this three-layer network, the above equations that represent the network function could be grouped as follows,

$$y_k(\mathbf{X}, \mathbf{W}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} \sigma\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)}\right) + b_k^{(2)}\right) \tag{2.21}$$

where the set of all weight and bias parameters have been grouped into a matrix $\mathbf{W}$. In this way, the model of the neural network is nested of non-linear functions from a set of input variables x_i to a set of output variables y_k controlled by the matrix $\mathbf{W}$ of adjustable parameters.

Usually, networks of a maximum of three layers are known as shallow neural networks; on the other hand, any network with more than three layers receives the name of *Deep Artificial Neural Network*.

Therefore, for a neural network with L layers, in which the j^{th} neuron accept a set of input responses generated by N previous neurons, the output of the l^{th} layer is estimated as

$$\mathbf{a}^{\mathbf{l}} = \sigma\left(\sum_{i=1}^{N} w_{ij}^{l} a_{ij}^{l-1} + b_j^{l}\right) \tag{2.22}$$

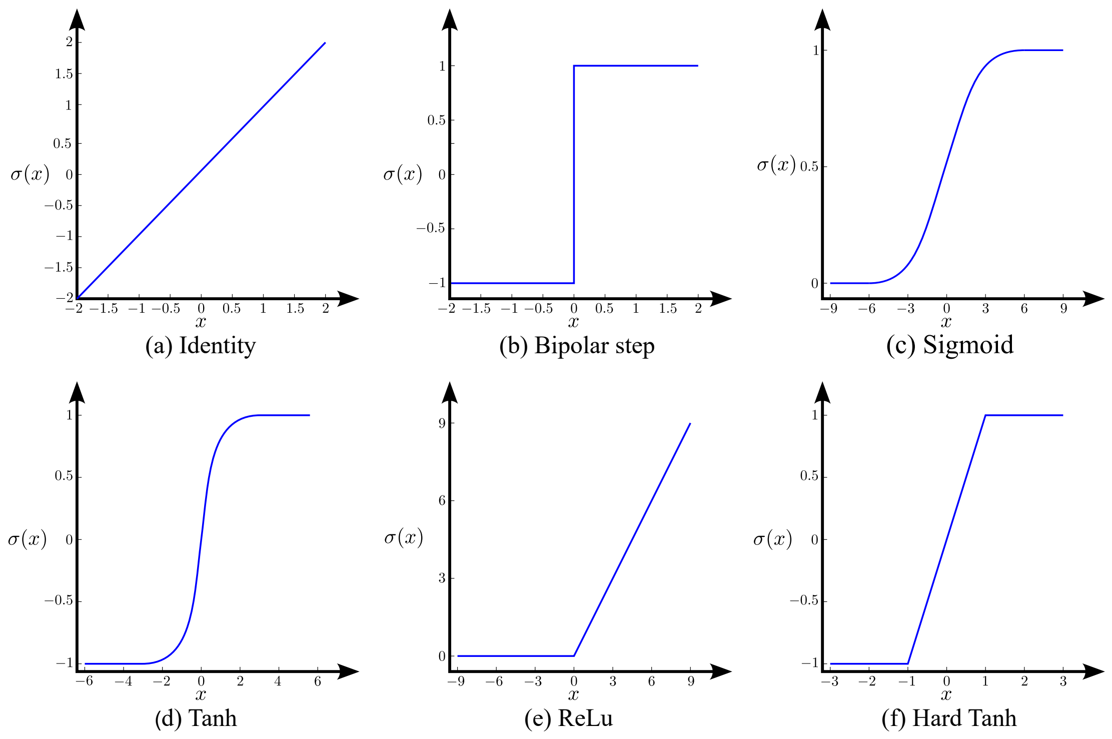
Then, expressed as matrix-vector notation,

$$\mathbf{a}^{\mathbf{l}} = \sigma\left(\mathbf{W}^T \mathbf{a}^{l-1} + \mathbf{b}\right) \tag{2.23}$$

where σ refers to any non-linear activation function of the neuron.

2.3.1.2. Activation functions

The activation function, as aforementioned, incorporates non-linearity to the neural network. That is, it allows to approximate a non-linear function and find relationships between the input variables and the target variables. In this way, activation functions could be classified as *linear activation function* and *non-linear activation functions.* Figure 2.8 shows some graphs of the commonly used activation functions in ANNs.



(a) Identity (b) Bipolar step (c) Sigmoid

(d) Tanh (e) ReLu (f) Hard Tanh

Figure 2.8: Common activation functions used in ANNs.

Linear activation function. The linear activation function, which is illustrated in Figure 2.8(a), is also known as "*identity function*" and it is defined as follows,

$$\sigma(x) = x \tag{2.24}$$

This function provides an output proportional to the input, which means that this function does not do anything to the activation potential (u in Equation 2.15). If used in hidden layers, no matter the number of layers a network has, it will be reduced to single layers because the output layer will be a linear transformation of the first layer. Additionally, since the derivative of this function is a constant, it is impossible to adjust the parameters w and b of the network for the training data ($\mathcal{X}$). Therefore, the identity activation function should not be used for hidden layers of ANN. However, the identity activation function could be used in the output layer of ANNs when the target is a real value.

No-linear activation functions. The activation functions in this group permit the model to create complex mappings between the network's inputs and outputs because they are derivative, which allows backpropagation, an algorithm that achieves the learning of the parameters of ANNs. Also, due to this characteristic of the functions, the stacking of multiple layers of neurons is possible because the output would now be a non-linear combination of input passed through multiple layers (Hecht-Nielsen, 1989).

The *bipolar step function*, whose graphical representation is shown in Figure 2.8(b) could be used to map a binary output at prediction time. The output produced by this function will be 1 when the neuron activation potential is greater than zero; 0 when the potential is also 0; and -1 when the potential is less than zero. The mathematical notation of this function is as follows,

$$\sigma(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases} \tag{2.25}$$

On the other hand, the outputs of the *sigmoid* activation function exist between $(0, 1)$, which is useful in performing computations that should be interpreted as probabilities; since probability exists only between the range of 0 and 1. The graphical representation of this function is illustrated in Figure 2.8(c), and it is mathematically described by the following expression,

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.26}$$

The *tanh* activation function, which is graphically shown in Figure 2.8(a), has a similar shape to the *sigmoid* function; however, its output range is between $(-1, 1)$. Its mathematical expression is written as,

$$\sigma(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.27}$$

However in the early implementation of ANN, the *sigmoid* and the *tanh* functions were often selected for incorporating non-linearity into the neural network.

In modern ANNs, scholars have replaced these functions with the *rectified linear unit* (ReLu) and *hard tanh* activation functions, whose graphs are shown in Figure 2.8(e) and Figure 2.8(f), respectively. The reason is that these functions are less affected by the problem of the vanishing gradient and typically show better convergence behavior (Lottes, 2021). The ReLU activation function performs a threshold; the input values that are less than zero are set to zero and it behaves as a linear activation for those values equal to or above zero (Equation. 2.28). On the other hand, the *Hard tanh* function lies within the range of -1 to 1 and is expressed by Equation 2.29.

$$\sigma(x) = \text{máx}(0, x) \tag{2.28}$$

$$\sigma(x) = \begin{cases} 1, & \text{if } \ x > 1 \\ x, & \text{if } -1 = x \leq 1 \\ -1, & \text{if } \ x < -1 \end{cases} \tag{2.29}$$

According to the description given above, the *identity* activation function must be used exclusively in the output layer of networks for standard regression problems. Respecting *ReLu* and *hard tanh*, they are most recommended to use in hidden layers of neural networks, and avoid using *sigmoid* and *Tanh* functions in these layers. Instead, *sigmoid* and *Tanh* functions could be implemented in the output layers of ANNs for binary classification and multilabel classification (Aggarwal, 2018).

In this thesis, the *ReLu* activation function was implemented in hidden layers of ANNs for feature classification and into CNNs (Section 2.3.2) for feature extraction. Additionally, since this work lies with multiclass classification, the output layer of the networks was provided with the number of neurons equivalent to the number of classes. Additionally, the *softmax* activation function was applied to the last activation vector ($\mathbf{a^l}$) , defined as follows,

$$\sigma_{softmax}(\mathbf{a^L}) = \frac{e^{\mathbf{a^l}}}{\sum_{j=1}^{C} e^{\mathbf{a^l}}} \tag{2.30}$$

where C is the number of classes.

The softmax function computes the probability distribution from a vector of real numbers. The length of the output vector is equivalent to C and their magnitude is between 0 and 1, with the sum of the probabilities being equal to 1. Therefore, the target class has the highest probability.

2.3.1.3. Loss functions

In this thesis, supervised ML was implemented. Therefore, we had the input data $\mathcal{X}$ and the actual labels $\mathcal{Y}$. In this way, the model's parameters were optimized with respect to a loss function $\mathcal{L}$ when training a neural network. The loss function $\mathcal{L}$ measures the difference between network's predictions and the labels. Consequently, the objective of the training procedure is to adapt the model parameters Θ to minimize the difference computed by $\mathcal{L}$. This is because the loss function always penalizes incorrect classifications, so that the network achieves adequate mapping between the input and the output.

The loss functions used depend on the architecture of the networks and the goal task. Therefore, they are grouped into regression loss functions and classification loss functions. Scholars have proposed a considerable number of loss functions, such as the ones documented in Terven et al. (2023). Since, in this thesis, the experiments focused on the classification of plants, we implement *Categorical cross-entropy loss* ($\mathcal{L}_{CCE}$), *dice loss* ($\mathcal{L}_{dice}$) and *focal loss* ($\mathcal{L}_{focal}$).

$\mathcal{L}_{CCE}$ measures the dissimilarity between the predicted probability distribution and the true distribution. This loss function is expressed as

$$\mathcal{L}_{CCE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij}) \tag{2.31}$$

where N is the number of samples, C is the number of classes, y is the true label, and p is the predicted probability of the true class.

The *dice loss* was used in the classification of pixels as a strategy to compensate the small ROIs that certain plant species might occupy in images; this is because this loss increases the IoU twice, promoting an easy localization of small ROIs into the images. The computation of dice loss is as follows,

$$\mathcal{L}_{dice} = 1 - \frac{2y\hat{y} + 1}{y + \hat{y} + 1} \tag{2.32}$$

where y refers to the ground truth label and $\hat{y}$ is the predicted value from the model.

Respecting the *focal loss*, it was used to compensate for the pixel imbalance. This is because plant species usually appear small in the images, predominating the background pixels. The focal loss addresses this issue by down-weighting the easy negative samples and up-weighting the hard positive samples. The mathematical notation of this function is as follows,

$$\mathcal{L}_{focal} = -\alpha_t(1 - p_t)^{\phi} log(p_t) \tag{2.33}$$

where $\alpha_t \in [0, 1]$ is a vector of class weights which is computed as the inverse class frequency from the dataset labels, p_t is a matrix of probabilities that each class has to be ground truth, and ϕ is the degree of modulating the pixels that are easy to classify (usually $\phi = 2$).

2.3.1.4. Training of neural networks

So far, the configuration of neural networks has been analyzed, encompassing input layers, hidden layers, output layers, parameters Φ, and differentiable activation functions. This network configuration facilitates the execution of a forward pass to predict an output based on input data, utilizing the current parameters Φ. Subsequently, the loss function estimates the error between predictions and labels. Here is when the network training comes in, involving the identification of suitable values for the parameters Φ to minimize the error calculated by the loss function. For the training of neural networks, the backpropagation algorithm and optimizer algorithms work together.

Backpropagation computes the gradient of a loss function concerning the weights of the network one layer at a time, iterating backward from the output layer until the input layer. In contrast, the optimizer algorithms perform the strategy for actualizing the weights using the gradients to optimize the loss function (Bishop, 2006).

Backpropagation. It is an algorithm for computing the gradient of the loss function concerning the parameters Φ of each layer of networks. It is simply an application of the chain rule for derivatives, and its definition can be expressed as follows,

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_j^l} \times \frac{\partial a_j^l}{\partial u_j^l} \times \frac{\partial u_j^l}{\partial w_{ij}^l} \tag{2.34}$$

where $\mathcal{L}$ is the error or cost function, a_j^l is the activated output of the neurons of layer l, u_j^l is the activation potential of neurons of layer l.

To clarify this algorithm, suppose the network is using the mean squared error loss function and *sigmoid* activation function (ec. 2.26), then the gradient of the error with respect to w_{ij} is as follows,

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial}{\partial a_j^l}(\frac{1}{2}(y - a_j^l)^2) \times \frac{\partial}{\partial u_j^l}\left(\frac{1}{1+e^{-u_j^l}}\right) \times \frac{\partial}{\partial w_{ij}^l}\left(w_{ij}^l a_j^{l-1} + b_j^l\right) \tag{2.35}$$

The gradient of the error with respect to weights is propagated backward to every neuron through each layer of the network. Subsequently, an optimizer algorithm actualizes the weights.

Optimizers. The networks' weights are actualized iteratively to learn the input data's features utilizing an optimizer algorithm. Choosing the correct optimizer could impact the evaluation accuracy and speed of training.

The traditional *gradient descent*, also called *full-batch learning* estimates the error gradient at the end of each epoch. One epoch is the set of iterations that goes through the entire dataset once. Nevertheless, it is not recommended to use gradient descent for extensive training datasets due to its time-consuming nature. This is because it processes the entire dataset for just one weight update during training. Moreover, when dealing with larger datasets, gradient descent consumes more memory, as it necessitates storing the complete dataset for training purposes. For that reason, in this thesis stochastic gradient descent (SGD) optimizer (Equation 2.36) and Adam optimizer (Equation 2.41) were implemented. In SGD, considering a dataset on n samples, the gradient of the error is updated in each iteration. SGD also allows dividing the whole dataset in *mini batches*, and then iterate over them and finally actualize the network parameter in a randomly selected *mini batch*, which results in a more efficient and feasible optimization procedure.

$$\mathbf{W}^{\tau+1} = \mathbf{W}^{\tau} - \eta \nabla E_n \tag{2.36}$$

where $\mathbf{W}^{\tau}$ are the weights in the iteration τ, η is the learning rate parameter and ∇E_n is the gradient matrix of the error. The learning rate guides the algorithm by determining the distance of each step and, consequently, the magnitude of parameter updates. It stands out as a crucial parameter requiring careful tuning. If the learning rate is too small, it results in insufficient learning progress. On the contrary, a huge learning rate can negatively impact convergence behavior and may even lead to divergence in the learning process (Bishop, 2006).

Respecting *Adam* optimizer, it combines the benefits of the optimizers RMSprop and SGD with momentum. Adam is an "adaptive moment estimator", which means, it computes individual learning rates for different parameters. It utilizes the squared gradients to adjust the learning rate similar to RMSprop, and it leverages momentum by incorporating a moving average of the gradient instead of the gradient itself, as seen in SGD with momentum. First of all, Adam optimizer computes an ongoing average of both gradients and squared gradients for each parameter in the model. Subsequently, these averages are employed to determine the updates for each parameter, as follows,

$$\mathbf{m_t} = \beta_1 \mathbf{m_{t-1}} + (1-\beta_1)\mathbf{g_t} \tag{2.37}$$

$$\mathbf{s_t} = \beta_2 \mathbf{s_{t-1}} + (1-\beta_2)\mathbf{g_t^2} \tag{2.38}$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m_t}}{(1-\beta_1^t)} \tag{2.39}$$

$$\hat{\mathbf{s}}_t = \frac{\mathbf{s_t}}{(1-\beta_2^t)} \tag{2.40}$$

where $\mathbf{g_t}$ is the gradient at time t, $\mathbf{m_t}$ and $\mathbf{s_t}$ are the first and second moments of the gradients, respectively, β_1 and β_2 are hyperparameters that control the decay rates of the moment estimates, and $\mathbf{g_t}$ is the gradient. Usually, $\beta_1 = 0.9$, indicating a long memory for the first moment, offering a reliable indication of the gradient's trend. Conversely, $\beta_1 = 0.999$, implying a shorter memory for the second moment, emphasizing the magnitude of the gradient (Murphy, 2022).

Finally, the weights update is as follows,

$$\mathbf{W}^{\tau+1} = \mathbf{W}^{\tau} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon} \tag{2.41}$$

where η is the learning rate, and ϵ is a small constant used to prevent division by zero, usually $\epsilon = 10^{-6}$.

2.3.2. Convolutional neural networks

CNNs are specialized for high-dimensional data, such as images and videos. Concerning image analysis, they could be used for classification, object detection and segmentation challenges. Also, they are used for both supervised and unsupervised learning approaches. In supervised learning, the inputs and their corresponding labels are known, while in unsupervised learning, the model seeks to estimate the underlying distribution of input data samples without knowledge of true labels for a given set of inputs (Khan et al., 2018).

A basic CNN is a compound of convolutional layers and fully connected layers (FCL), as shown in Figure 2.9. Convolutional layers are specialized in feature extraction, also known as feature learning (Murphy, 2022). A CNN could have multiple convolutional layers, which gives the depth of the network. On the other hand, the FCL are used for classification and follow the typical architecture of an ANN, already covered in the last sections.

The CNN receives input data organized in a 2D grid structure $I \in \mathbb{R}^{h \times w}$, where h is the height and w is the width of the structure, or also a CNN could receive a 3D grid structure $I \in \mathbb{R}^{h \times w \times d}$, where d is the depth of the structure. In the case of RGB images, the values at each grid point are known as pixels, representing specific spatial locations within the image ($I_{hw}(i, j)$). Since CNNs need to be fed with an array of values, an RGB image could be transformed to a 3D array (structure) of size $I \in \mathbb{R}^{h \times w \times d}$, where (h, w) is the spatial dimension of the image and d is the depth, representing the RGB channels (Aggarwal, 2018).

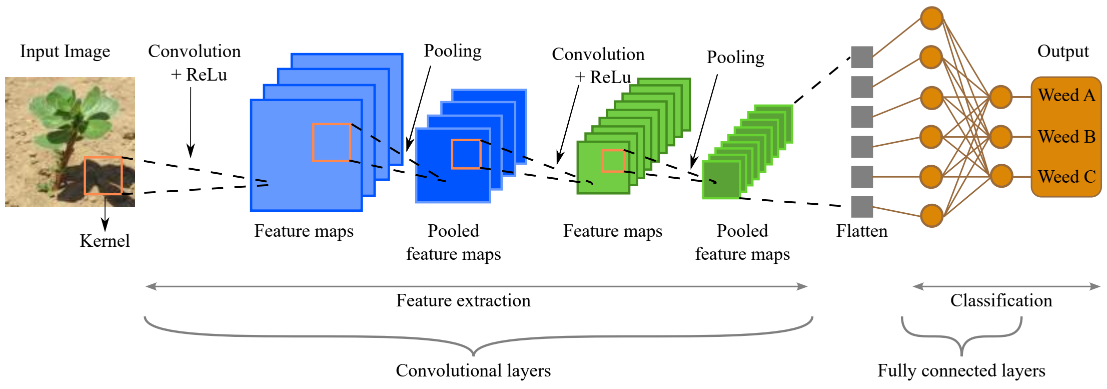


Figure 2.9: A simple architecture of a classification CNN. Convolutional layers are specialized in feature extraction, and their number determines the depth of the network; whereas, the fully connected layers are in charge of classification.

A convolutional layer of a CNN commonly involves the operations convolution, non-linear activation function (usually ReLU) and pooling. Therefore, as the pixel intensities of primary colors enter the first layer of a CNN, the two dimensions capture spatial relationships, while the third dimension accounts for independent properties along channels. After $I_{hwd}(i, j)$ is operated by the first convolutional layer, it produces *feature maps* ($M \in \mathbb{R}^{h \times w \times d}$). In this way, every convolutional layer, as we move forward into layers, produces an array of feature maps with reduced spatial size M_{hw} but with major depth (M_d) compared to the previous layer (Aggarwal, 2018). In simpler terms, the network constructs a hierarchical representation of the input. For instance, in the context of this thesis, where images of weed species serve as input, the initial layers depict basic features like edges, subsequent layers capture more intricate features like corners, and deeper layers can identify abstract features such as leaves or stems. All these are achieved through the training process. Further details on the operations within convolutional layers will be discussed below.

2.3.2.1. Input image

In the Section 1.1.3, it was mentioned the spatial resize, background removal, enhancement, denoising, and augmentation as pre-processing techniques carried out over input images for training CNNs. Nonetheless, it is mandatory to reduce the range of the input values (pixels) over which the backpropagation algorithm works, so that the variables vary over a small range, which facilitates the training. Mean-subtraction and normalization are the frequent algorithms applied over input images. Consider an image $I(i,j) \in \mathbb{R}^{h\times w\times d}$, then mean-subtraction and normalization are computed according to Equation 2.42 and Equation 2.45, as follows

$$\hat{I}(i,j) = I(i,j) - \mu \tag{2.42}$$

$$\mu = \frac{1}{hw}\sum_{i=0}^{h-1}\sum_{j=0}^{w-1} I(i,j) \tag{2.43}$$

$$\sigma^2 = \frac{1}{hw}\sum_{i=0}^{h-1}\sum_{j=0}^{w-1}[I(i,j)-\mu]^2 \tag{2.44}$$

$$x(i,j) = \frac{\hat{I}(i,j)}{\sqrt{\sigma^2}} \tag{2.45}$$

where $\hat{I}(i,j)$ is the mean-substracted image, μ is the mean of the image $I(i,j)$, σ^2 is the variace of the image $I(i,j)$ and $x(i,j)$ is the normalized image.

2.3.2.2. Convolution layers

A 2d-convolution operation is executed using a kernel f over the input. The most often used filters are of size $(7\times 7), (5\times 5)$, (3×3) and (1×1). The kernel traverses the entire input, conducting a dot product between its values and the corresponding values of the input at each position.

The mathematical expression of a 2d-convolution is expressed as

$$[\mathbf{f} \circledast \mathbf{I}](i,j) = \sum_{u=0}^{h-1}\sum_{v=0}^{w-1} f_{(u,v)} I_{(i+u,j+v)} \tag{2.46}$$

where f is a 2d kernel of size (h,w). The output from a convolution $\mathbf{M} = \mathbf{f} \circledast \mathbf{I}$ is named *feature map.*

The Figure 2.10 is a representation of the convolutions operation among a kernel of size (3×3) and an input of size (5×5). Always, a $f_h \times f_w$ kernel over an image of size $I_h \times I_w$ produces an output of size $(I_h - f_h + 1)\times(I_w - f_w + 1)$.

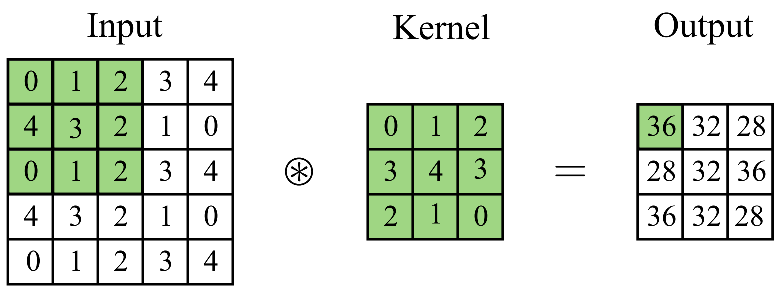


Figure 2.10: Illustration of a 2d-convolution with a (3×3) kernel and (5×5) input. Notice that the output has been reduced by $f - 1$ along the height and the width concerning the input.

The convolution illustrated in Figure 2.10 is called *valid convolution* because the kernel always is applied in "valid" positions of the input feature map. However, this kind of convolution produces an output feature map reduced by $f - 1$ along the height and width with respect to the input. This type of size reduction is not desirable in general, because it tends to lose some information along the borders of the image or feature map, in the case of hidden layers. That is, the contributions of the pixels on the borders of the image or feature map will be under-represented in the next hidden layer, which is undesirable.

This problem can be resolved by using zero-padding. Zero-padding means adding pixels of value zero all around the borders of the input image or feature map, as illustrated in Figure 2.11. Importantly, the border sections do not influence the ultimate dot product as their values are set to 0. To estimate the amount of padding to ensure that the output retains the same size as the input depends on the kernel size, and it is denoted by $p = \frac{f-1}{2}$. In this case, the convolution is referred to as *same convolution*. The output size of *same convolution*, if the input has size $I_h \times I_w$ and a kernel of size $f_h \times f_w$ is expressed as

$$(I_h + 2p_h - f_h + 1) \times (I_w + 2p_w - f_w + 1) \tag{2.47}$$

So far it has been covered convolution operation when the kernel slides one step along the horizontal or vertical position of the input. This step is referred to as the *stride* of the convolution filter. However, the consequence of using a *stride* of 1 is that each spatial location of the feature map $M(i, j)$ will be similar in value to its neighboring because the regions covered by the kernel overlap

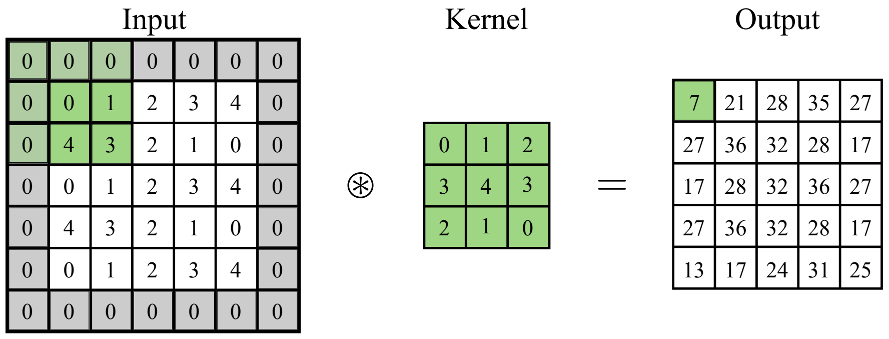
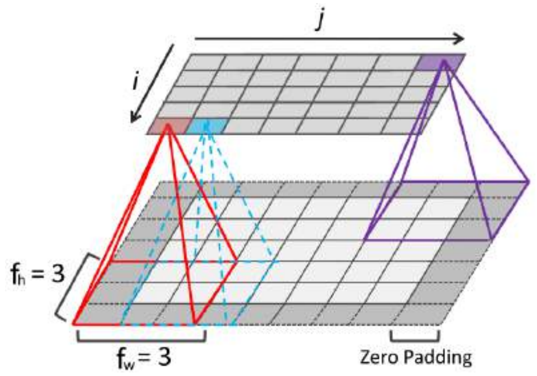


Figure 2.11: Same convolution representation using padding of one. The output is the same size as the input.

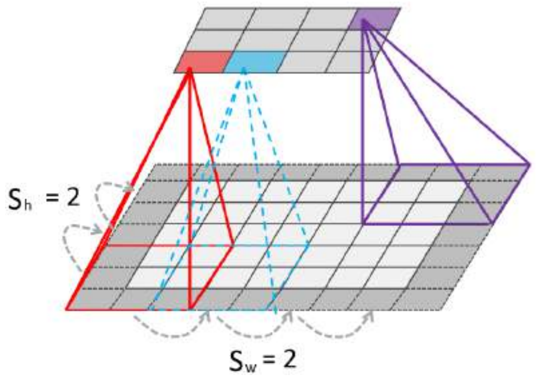
for their computation (Murphy, 2022). Nonetheless, the stride value could be modified to a value of two, and it is rare to use strides more than 2 in normal circumstances. Larger strides reduce computation costs and reduce overfitting if the spatial resolution of the input is unnecessarily large (Aggarwal, 2018). Strides have the effect of rapidly increasing the receptive field of each feature in the hidden layer (the size of the region in the input that produces the feature) while reducing the spatial size of $M_h \times M_w$ in the specific layer. Large receptive fields permit to capture complex features in a larger spatial region of the image (Khan et al., 2018). For instance, Figure 2.12(a) represents a *same convolution* (zero padding) among a 5×7 input and a 3×3 kernel, which produces a 5×7 output. In this scenario, the receptive field (input) and output (feature map) are of the same size. On the other hand, Figure 2.12(b) shows a convolution with an input of size 5×7 and a kernel of size 3×3, but with a stride of two. In this scenario, the output is of size 3×4, which means that the feature map is produced by a large receptive field. In general, if the input has size $I_h \times I_w$, kernel of size $f_h \times f_w$, padding is used of size p_h and p_w, and stride of sizes s_h and s_w, the output size is determined as

$$\left(\frac{I_h + 2p_h - f_h + s_h}{s_h}\right) \times \left(\frac{I_w + 2p_w - f_w + s_w}{s_w}\right) \tag{2.48}$$

Scenarios covered above, in Figure 2.10 and Figure 2.11 correspond to 1d-input, such as a gray image. Nonetheless, in practice, we face inputs of multiple channels, such as an RGB image that has three channels. In this issue, the kernel



(a)



(b)

Figure 2.12: Illustration of padding and stride in 2D convolution. (a) *Same convolution* among a 5×7 input and a 3×3 kernel, which produces a 5×7 output, and (b) convolution with an input of size 5×7 and a kernel of size 3×3, but with a stride of two; the output is of size 3×4. Adapted from Murphy (2022).

must have the same number of channels as the input data to produce an output of one channel. Therefore, the kernel f is an nd weight matrix or tensor. Each spatial grid location of the output (feature map) is mathematically expressed as follows

$$M(i,j) = b + \sum_{u=0}^{h-1} \sum_{v=0}^{w-1} \sum_{c=0}^{C-1} w_{(u,v,c)} I_{(si+u,sj+v,c)} \tag{2.49}$$

where s is the stride and b is the bias term.

To comprehend a third-dimension convolution, Figure 2.13 illustrates an input of 3 channels convolved with a filter of similar channels. In general, the shape of the output (feature map) from a third-dimension convolution is defined by the expression

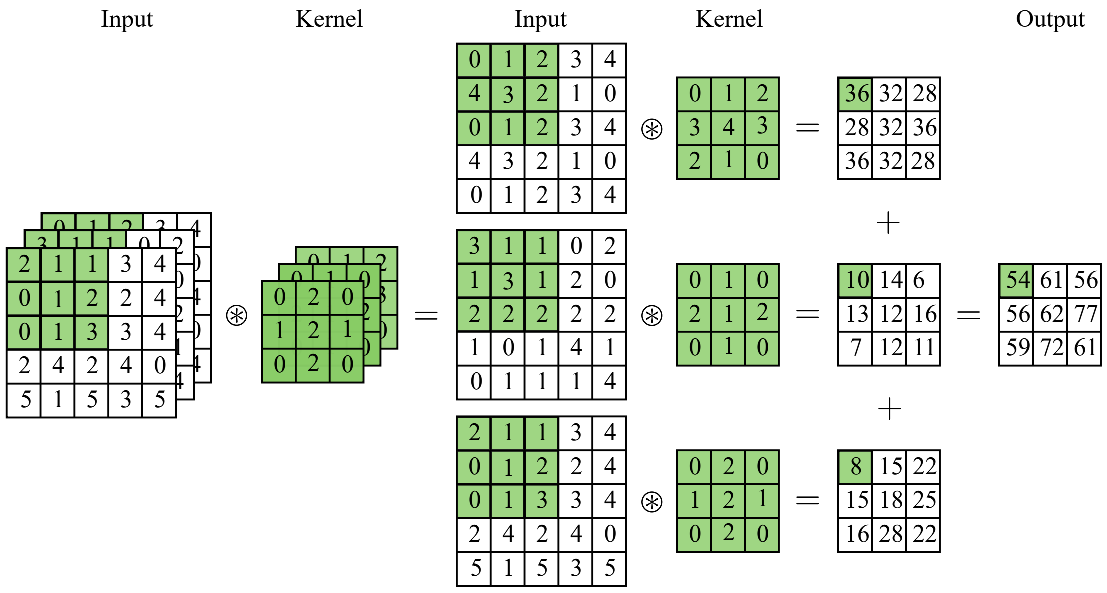


Figure 2.13: Ilustration of a 2d-convolution performed over an input of 3 channels.

$$\left(\left(\frac{I_h + 2p_h - f_h + s_h}{s_h}\right) \times \left(\frac{I_w + 2p_w - f_w + s_w}{s_w}\right) \times nf\right) \tag{2.50}$$

where nf is the number of filters.

2.3.2.3. Activation function

The feature map M derived from each convolution operation in every convolutional layer is provided non-linearity with an activation function. The application of ReLu in these layers is not different from how it is applied in an ANN. An activation function thresholds each spatial location of the feature map M without changing the spatial dimensions $(M_h \times M_w)$ of the feature map, because it is a simple one-to-one mapping of activation values. Early CNNs implemented distinct activation functions, such as *sigmoid* and *tanh*. Nevertheless, Krizhevsky et al. (2012) demonstrated that employing the ReLu offers significant benefits compared to other activation functions, showcasing superior speed and accuracy.

The heightened speed is intricately linked to accuracy, as it enables the utilization of deeper models and trains them for more epochs.

2.3.2.4. Pooling operation

The normalized feature map produced by both the convolution operation in every convolutional layer is a concentration of extracted features from the input. Since the feature maps are carried out by kernels, the architecture at this stage knows the exact position of the features in the input. Then, *pooling operation* congregate the features providing "translation invariance"to the model. That is, pooling gives the features that help the models to decide whether or not an object is in the input and the convolution operation provides the spatial localization of the object anywhere in the input.

The pooling operation works on small grid regions of size $p_h \times p_w$ over features maps and produces *pooled feature maps* with the same depth (number of channels) of the input feature maps. As the pooling grid window slides over the input feature maps, it could return either the maximum value or the average, receiving the name of *max-pooling* or *average-pooling*; an instance of this operation is shown in Figure 2.14. Therefore, the pooling operation drastically reduces the spatial dimensions of the input feature maps. In this way, the dimensional size of the output pooled feature map is defined as

$$\left(\frac{M_h - p_h + s_h}{s_h}\right) \times \left(\frac{M_w - p_w + s_w}{s_w}\right) \tag{2.51}$$

where M_h and M_w are the height and the width, respectively, of the input feature maps, p_h and p_w are the height and the width of the pooling window (usually of value 2) and s_h and s_w are the stride of the pooling window (usually of value 2) in the vertical and horizontal direction.

2.3.2.5. Remarks of CNNs

So far, the basic operation involved in a convolutional layer has been explained. However, CNNs could be implemented in vision challenges such as classification, segmentation, and detection. All these approaches use a set of strategically stacked convolutional layers with the same purpose, for feature detection and

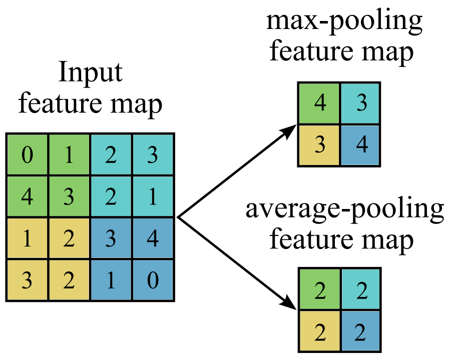


Figure 2.14: Ilustration of max-pooling and average-pooling operation using a 2 × 2 window over a 4 × 4 feature map.

learning (Murphy, 2022). Similar to ANNs, this can be achieved during the training of the models. Training a convolutional layer means adapting the values of the kernels and bias value (the parameters) of each convolutional layer to the input dataset (Skansi, 2018).

In the case of classification architectures, such as the one represented in Figure 2.9, the deeper pooled feature maps are flattened and subsequently an ANN classifies the features, either into binary or multiclass. For classification, it has been developed a considerable number of architectures, and some of them were already listed in Table 1.5 of section 1.1.3. Concerning segmentation networks, they usually benefit principally from the convolutional layers, and further deconvolution operations are performed to classify the pixels of the input into classes. Finally, detection architectures use convolutional neural layers complemented with specialized architectures, such as Region proposal Networks (RPN) (Ren et al., 2015) to localize objects in the feature maps, which are then separately classified by ANN.

2.3.3. Transformers

Transformers are deep learning models proposed for the first time by Vaswani et al. (2017) for machine translation. Attention mechanisms are which characterize a transformer architecture; they allow the model to handle long-range correlations between the input-sequence items.

The original transformer (Vaswani et al., 2017) consists of two main building blocks; an encoder and a decoder. An encoder module has two sub-layers; the first one is a multihead self-attention mechanism, and the second is a position-wise fully connected feed-forward network. The encoder generates an embedding vector $\mathbf{Z} = (\mathbf{z_1}, ..., \mathbf{z_n})$ from an input representation sequence $(\mathbf{x_1}, ..., \mathbf{x_n})$. On the other hand, the decoder contains three sub-layers, in addition to the two sub-layers of the encoder, it inserts a third sub-layer, which performs multi-head attention over the embedding vector $\mathbf{Z}$ to generate an output sequence $(\mathbf{y_1}, ..., \mathbf{y_n})$.

Later transformers designed for processing images follow the same principle as that proposed by Vaswani et al. (2017). That is, similar to text-based transformers, they also work with input representation sequences that are processed by an encoder. Figure 2.15 visually illustrates a transformer encoder, which contains two sub-layers within the block, preceded by an embedding patches block.

To clarify first the embedding patches block, let's consider an input image with height H, width W, and C channels. Then, pachifying the image into patches height and width size both as p, the image is splitted into a sequence of $m = \frac{HW}{p^2}$ patches. Subsequently, each patch is flattened to a vector $\mathbf{x_i} \in \mathbb{R}^{1 \times Cp^2}$. In this way, image patches can be treated similarly to tokens in text sequences by transformer encoders. Then, the flattened vectors are grouped into a matrix $\mathbf{z_0} = (\mathbf{x_1}, ..., \mathbf{x_n})$ where $\mathbf{z_0} \in \mathbb{R}^{m \times Cp^2}$. Following this, $\mathbf{z_0}$ is linearly transformed by multiplying it with three distinct embedding matrices $E \in \mathbb{R}^{Cp^2 \times D}$. D is a constant latent vector that keeps its size through the layer of the transformers, and it is related to the number of parameters of the model and the performance. Vision transformer (Dosovitskiy et al., 2021) uses D of size 768, 1024, and 1280. Embedding results in the matrices $Q \in \mathbb{R}^{m \times D}$ (query), $K \in \mathbb{R}^{m \times D}$ (key), and $V \in \mathbb{R}^{m \times D}$ (value). Figure 2.16 shows the visual representation of embedding four patches of an image to obtain the Q, K and V matrices.

Then, inside the encoder, the attention mechanism is performed. The attention mechanism is expressed as follows

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.52}$$

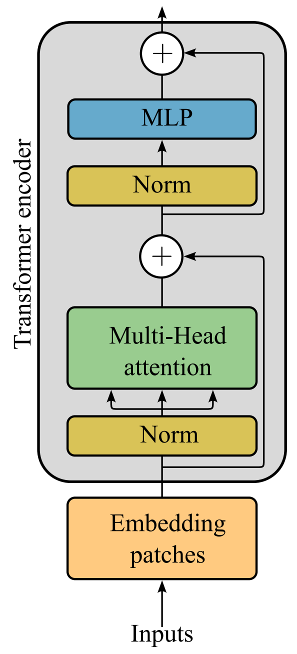


Figure 2.15: The schematic representation of a transformer encoder. Adapted from Dosovitskiy et al. (2021).

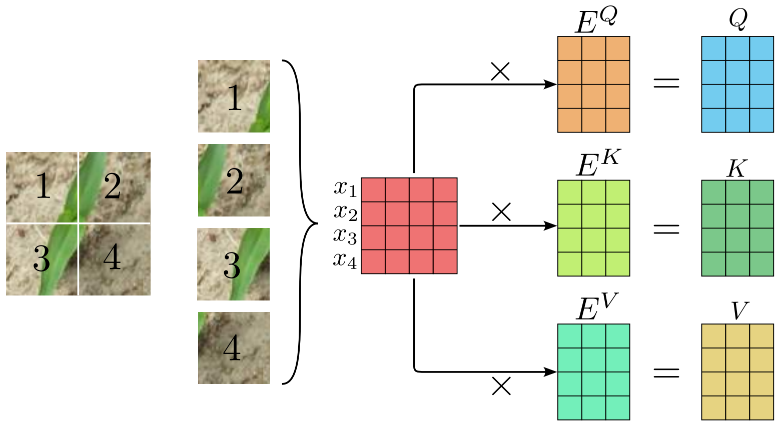


Figure 2.16: The embedding patching representation utilized by transformers designed for processing images.

where Q, K, and V are the query, key, and value matrices, K^T is the transpose of matrix K and d_k is the dimension of the vectors in query and key matrices. The expression $softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$ is named the *attention matrix*, and gives the weights of the values. The following Figure 2.17 represents a simple attention mechanism operation in a transformer.

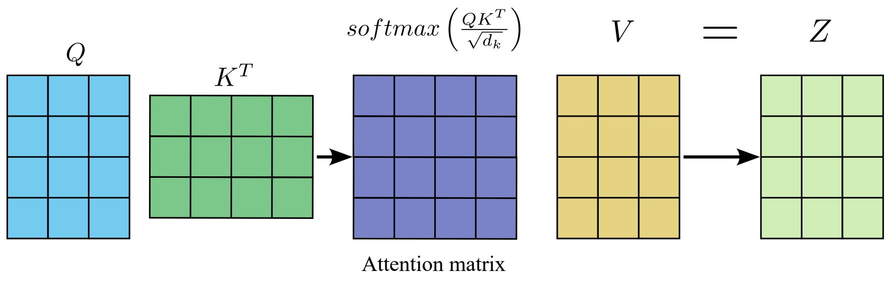


Figure 2.17: Representation of the attention mechanism in a typical transformer.

In practice, having the same $Q \in \mathbb{R}^{m\times D}$, $K \in \mathbb{R}^{m\times D}$, and $V \in \mathbb{R}^{m\times D}$, it is better to use different subspaces representation of them for analyzing them in parallel in a *miltihead attention* mechanism. Therefore, every subspace is represented as $Q \in \mathbb{R}^{m/h\times D}$, $K \in \mathbb{R}^{m/h\times D}$, and $V \in \mathbb{R}^{m/h\times D}$, where $h_i(i = 1, ..., h)$ is the number of heads. Finally, each subspace of Q, K and V is anew multiplied with distinct weight matrices W_i. The individual outputs from the sub-attention are concatenated and then transformed using another learned linear projection represented W^O to produce a final output. In this way, multihead attention is expressed as

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_h)W^O \\ where \quad head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.53}$$

where $W_i^Q \in \mathbb{R}^{D\times m/h}$, $W_i^K \in \mathbb{R}^{D\times m/h}$, and $W_i^V \in \mathbb{R}^{D\times m/h}$ and $W^O \in \mathbb{R}^{m\times D}$. Multihead attention gives models the ability to jointly attend multiple positions.

Finally, the encoder and decoder of transformers include MLP networks, as described previously, applied to the multihead attention output. Typically, they involve two linear transformations with a ReLU activation function, as follows

$$MLP(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.54}$$

The training of a transformer network involves adjusting the parameters of both the embedding matrices E and the weight matrices W in both the multihead attention and the MLP network.

2.4. Evaluation metrics

A performance metric serves to assess the model post-training, gauging its ability to generalize to novel data and make precise predictions. These metrics additionally facilitate comparisons among various models or configurations to identify the most effective one. In this thesis, classification and segmentation approaches were performed; therefore, the metrics accuracy, precision, recall, and F1 score were implemented. Additionally, the metrics proper for segmentation models DSC intersection over union, and mIoU were used to evaluate the models for segmentation.

The confusion matrix is used to define the performance of a classification algorithm. In this case, Figure 2.18 is a confusion matrix representation for a binary classification model, supposing plant and non-plant in the case of this thesis. Supposing the green circles are pixels of plants (foreground) and the red circles are pixels of the non-plant class (background). In this way, the plant pixels correctly classified are those that are in the green area, so they are the true positives (TP), whereas the correctly classified pixels as non-plant are called true negatives (TN) that are in the blue area. Since a classifier model also produces miss-classifications, that is, false positives (FP) and false negatives (FN). FP are those pixels that are predicted as plant pixels that actually belong to the class non-plant. In contrast, FN are those pixels that are predicted as non-plant class but actually, they belong to the class plant.

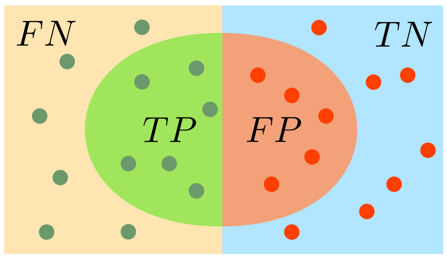


Figure 2.18: Confusion matrix representation from binary classification problem.

Accuracy is the ratio between the number of correct classified samples to the total number of samples. This metric works well if the number of samples belonging to each class is equal. In terms of the confusion matrix can be expressed as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.55}$$

Precision measures the ability of the model to identify targets; that is, the accuracy of positive predictions. A high precision indicates that the model produces few false positives, thereby ensuring the reliability of its predictions. This metric is defined as

$$Precision = \frac{TP}{TP + FP} \tag{2.56}$$

On the other hand, the recall metric, also known as sensitivity, assesses the model's ability to detect all positive instances within the dataset. A high recall value suggests that the model has a reduced number of false negatives, signifying its capacity to accurately identify a majority of positive instances. In terms of confusion matrix, it can be formulated as

$$Recall = \frac{TP}{TP + FN} \tag{2.57}$$

Respecting F1 score, it represents the overall performance of a model providing a unique value by the combination of precision and recall. In other words, it is the harmonic mean of the precision and recall. A superior F1 score suggests an improved equilibrium between precision and recall, while a lower F1 score implies that the model may have high precision or recall but not in both. This metric is especially valuable in scenarios with imbalanced class distribution or when equal importance is assigned to precision and recall (Terven et al., 2023). The expression of this metric is

$$F1\,score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.58}$$

For evaluating the performance of segmentation models in this thesis the IoU, mIoU and DSC were used. IoU serves as the fundamental metric to quantify the overlap between predicted and ground truth regions in object detection and segmentation. A higher IoU value indicates a better alignment between the predicted and actual regions, reflecting a more accurate model. This metric is visually represented in Figure 2.19(a), and is expressed as follows

$$IoU = \frac{TP}{FP + TP + FN} \tag{2.59}$$

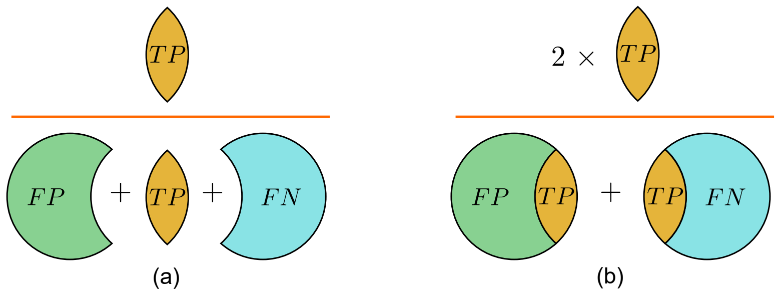


Figure 2.19: Graphical representation of the metrics IoU and DSC. (a) IoU and (b) DSC.

Therefore, the mIoU metrics give the average segmentation performance of a model, quantifying the N number of classes. It is expressed as

$$mIoU = \frac{1}{N} \sum_{i=1}^{N} IoU_i \tag{2.60}$$

Finally, DSC, also known as F1 score in classification tasks, is a measure of the similarity between two sets, A and B. The coefficient ranges from 0 to 1, where 1 indicates that the two sets are identical, and 0 indicates that the two sets have no overlap. Figure 2.19(b) gives a visual representation of this metric. Then, the following expression defines it in terms of the confusion matrix

$$DSC = \frac{2TP}{FP + 2TP + FN} \tag{2.61}$$

Chapter 3

Methodology

As stated, this thesis aims to develop a vision system based on DL models to detect weeds in natural corn fields and implement it on a sprayer machine for real-time herb control. Therefore, to achieve the goal, we first developed the vision system for crop/weed detection. Subsequently, a mechatronic platform was designed; which is commanded for the vision system, and finally, both components were field-evaluated. Figure 3.1 provides a broad overview of the methods follow in this thesis.

Any vision system for object recognition requires a dataset to utilize features for subsequent classification. Consequently, for the development of the vision system, a large dataset was created and annotated, which is detailed in Section 3.1.1. Concerning crop/weed detection, it involves localizing single or multiple instances of plant species in an image and classifying them into their respective categories. In this study, the classes Crop, NLW, and BLW were proposed to group the plant species found in a typical corn field. However, since weed localization could be achieved using classical thresholding algorithms to isolate the plants in an image, followed by classification using either shallow classifiers or ANNs, we first explored a classification approach using classical descriptors, shallow classifiers, and DL classifiers, as proposed in Section 3.1.2. Additionally, detection also could be achieved by applying semantic segmentation of the plants with CNNs and then localizing the ROIs separately. Therefore, Section 3.1.3 attacks the problem using region-based CNNs and U-Net-like architectures. Section 3.1.4 addresses the problem with the cooperation of segmentation CNNs and classification CNNs. Finally, in Section 3.1.5, another vision strategy addressing the problem using later transformer architectures is covered.

At stage two of the methods, the mechatronic platform commanded by the weed-vision system was developed, as detailed in Section 3.2. This development involved mechanical, electrical, and hydraulic design, culminating with the vision system release. In the final stage, the entire intelligent weed control system was evaluated under an authentic cornfield, as described in Section 3.3.

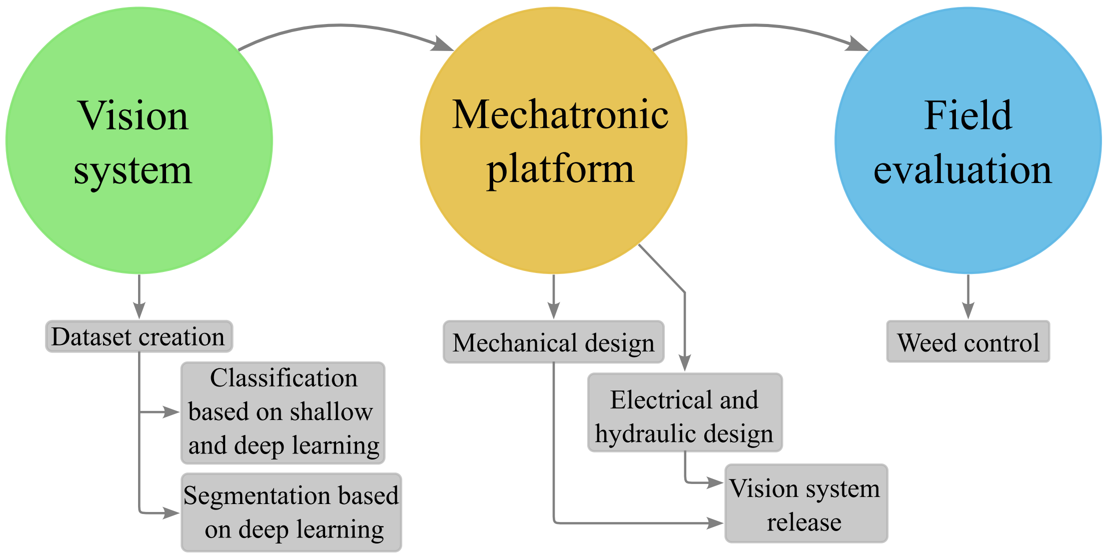

Vision system
Mechatronic platform
Field evaluation
Dataset creation
Classification based on shallow and deep learning
Segmentation based on deep learning
Mechanical design
Electrical and hydraulic design
Vision system release
Weed control

Figure 3.1: An overview of the general methodology used to achieve the thesis goal.

3.1. Vision system development

3.1.1. Dataset creation and description

The image dataset of plant species was collected considering natural field conditions typical of a corn field. To capture diverse conditions, five corn plots of $0.5ha$ each were established in distinct locations throughout the state of Aguascalientes, Mexico. Two of these corn plots were established in 2020, two in 2021, and one in 2022, during the spring-summer cycle. In the plots established in 2020 and 2021 RGB images were collected, and in the plot of 2022, it were captured both RGB and multispectral images. Figure 3.2 shows a sequence of a corn plot used for image acquisition in this thesis, from soil preparation, sowing, and two growth stages of the plants.

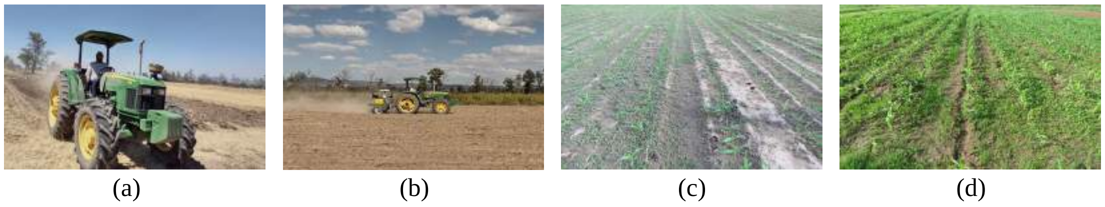
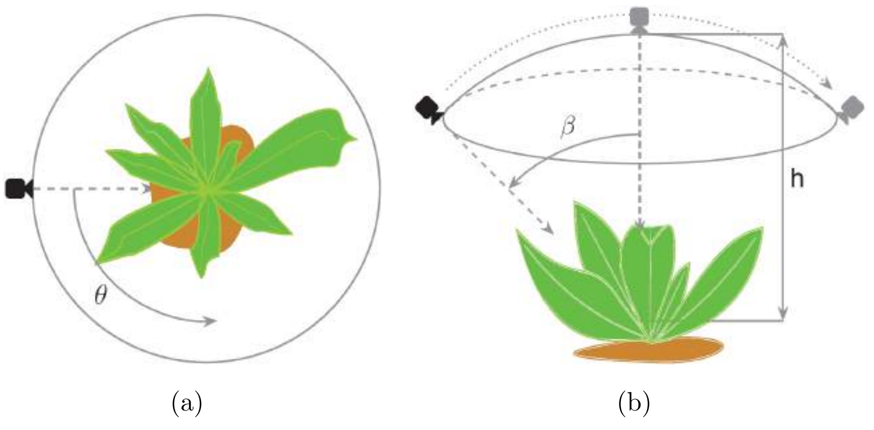


Figure 3.2: Sequence images from a corn plot used for image acquisition. (a) Soil tillage, (b) corn sowing, (b) early growth stage of plants and (d) high density of plants.

The process of obtaining images took place at intervals of five days, specifically during the developmental phase when corn plants had between two and seven leaves. Figure 3.3 displays the camera positions utilized for image capture. Here, $\theta \in [0, 2\pi]$ represents the camera's rotational position relative to the target (Figure 3.3(a)), and $\beta \in [-\pi/4, \pi/4]$ signifies the lateral orientation of the camera (Figure 3.3(b)). When $\beta = 0$, the camera provides a top-down view. Additionally, h denotes the distance between the camera and the target base, with a maximum value of approximately $1.50m$ and a minimum value essential for capturing either a corn plant or weed. For the RGB images, the devices used were a Canon PowerShot Sx60HS 16.1-megapixel camera with a resolution of 4608×3456, and smartphones, which provided images of sizes 1600×720, and 2460×1080 pixels. On the other hand, multispectral images were captured with a MicaSense camera (RedEdge M), which gives an image resolution of 1280×960 pixels. In Figure 3.3(c) a visualization of a capture is depicted.

Our dataset encompasses various forms of variability. Images contain single-plant and, more prominently, multi-plant with different species of weeds and several instances of the crop as a function of the zoom (distance h). The dataset also includes sunlight variability as image captures were conducted under sunny and cloudy conditions and at different times of the day, such as morning, noon, and afternoon. Additionally, some captures were taken immediately after rainfall events. Moreover, given the strategy followed to acquire images, occlusion and overlap of foliage are introduced, especially in the late growth stages of the plants.



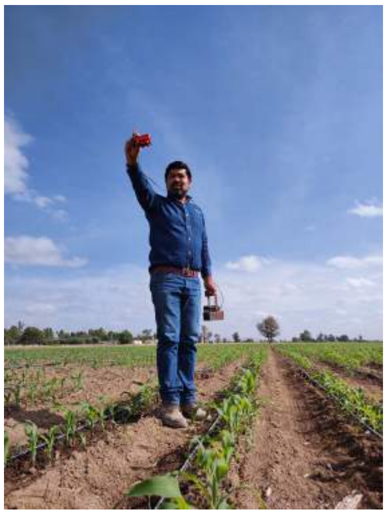


Figure 3.3: Camera configuration for capturing images and visualization of the scene. (a) top view, (b) side view and (c) visualization of a capture.

Background variations encompass stones, soil attributes related to humidity levels and texture, as well as remnants like straws from previous crops, among other elements.

A sample of images that integrates the dataset is provided in Figure 3.4. The first row, Figure 3.4(a), displays images of individual plants. Moving to the second row (Figure 3.4(b)), there are images of multiple plants with overlapping leaves, occlusion, and variations in soil appearance. The last row, Figure 3.4(c), illustrates the representation of multiple small plants resulting from the maximum capture distance ($h = 1.5m$).

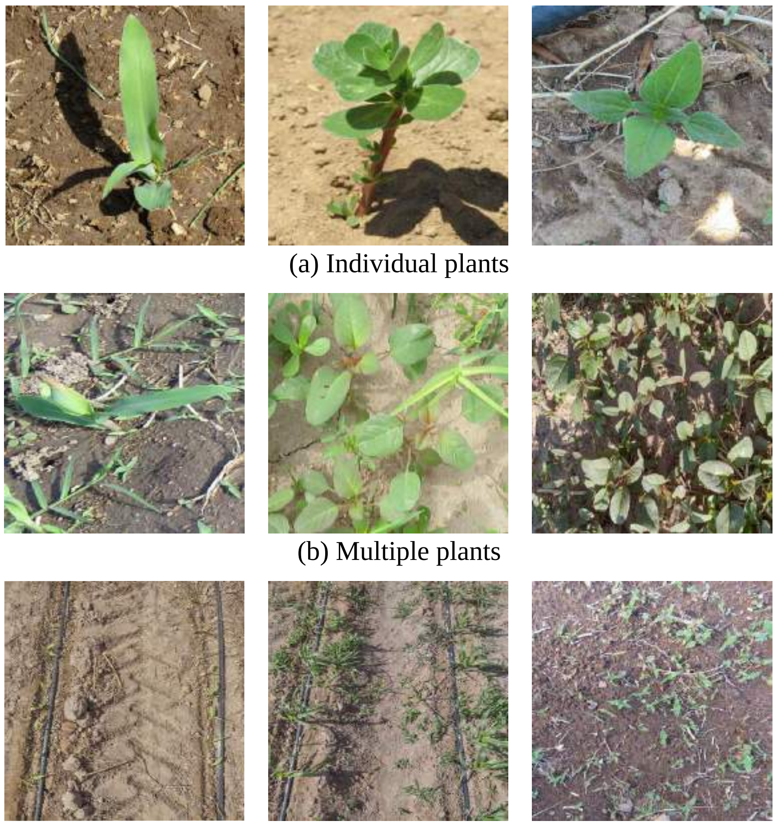
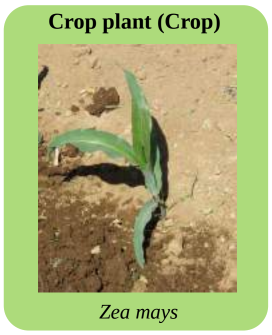


Figure 3.4: Sample of images that are in our dataset.

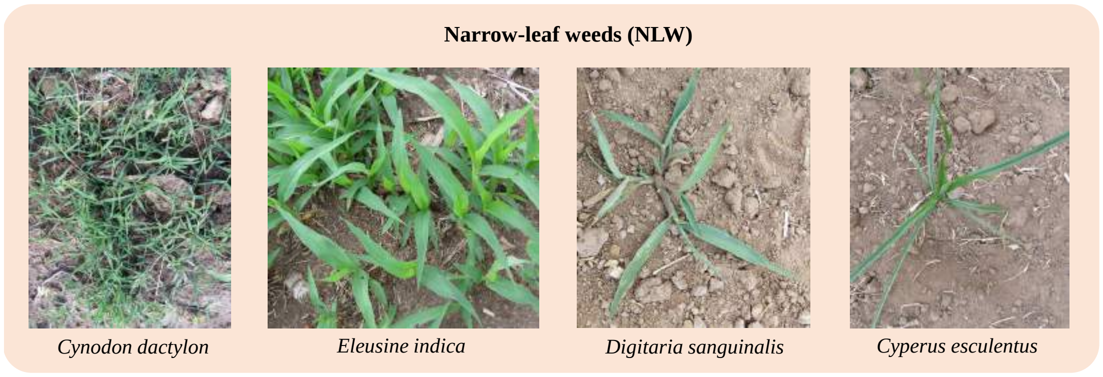
3.1.1.1. Annotation of RGB images

In total, it was captured 15,885 RGB images. After carefully analyzing the plant species in the images, it was found the following predominant plant species; Crop plant (*Zea mays*); and the weeds, *Cynodon dactylon*, *Eleusine indica*, *Digitaria sanguinalis*, *Cyperus esculentus*, *Portulaca oleracea*, *Tithonia tubaeformis*, *Amarantus spinosus*, and *Malva parviflora*. Images containing a sample of the species are provided in the following Figure 3.5.

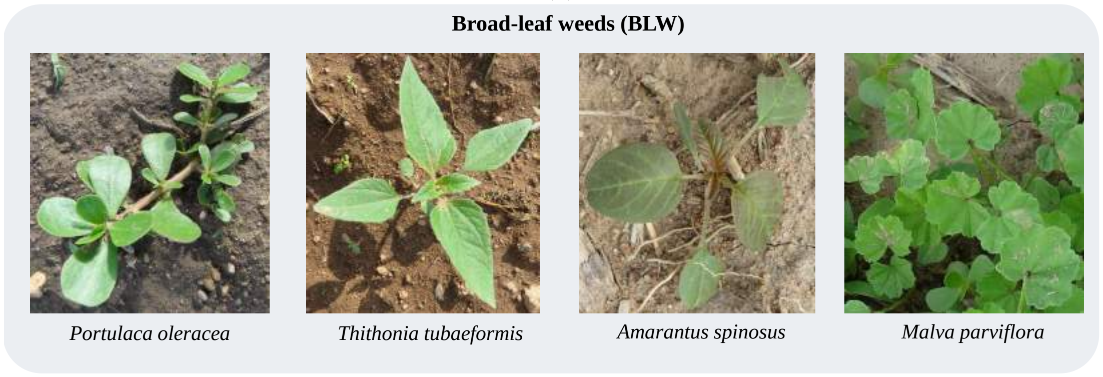
Subsequently, the plant species were manually annotated at the pixel level using the tool VGG Image Annotator (Dutta and Zisserman, 2019). This involved tracing carefully a polygon around the contour of most plants in the image,



(a)



(b)



(c)

Figure 3.5: Plant species that integrate the experimental dataset. (a) A crop plant sample, (b) a sample plant of every NLW species, and (c) a sample plant of every BLW species.

ensuring that soil pixels were consistently excluded. The species name was then assigned to the pixels within the polygon. Figure 3.6(a) shows a visualization of polygons enclosing the plants in an image. Whereas Figure 3.6(a) illustrates the image with a color assigned to specific pixels. In this case, green regions represent corn pixels, red regions belong to pixels of narrow-leaf weeds and blue pixels denote bread-leaf weed pixels.

Following this strategy, a total of $10,575$ images were annotated. t is worth mentioning that the number of plant species labeled is known under this scenario, and the soil pixels were indirectly annotated. Figure 3.7 summarizes the dataset distribution. This graph depicts the number of plant species and their equivalent percentage.

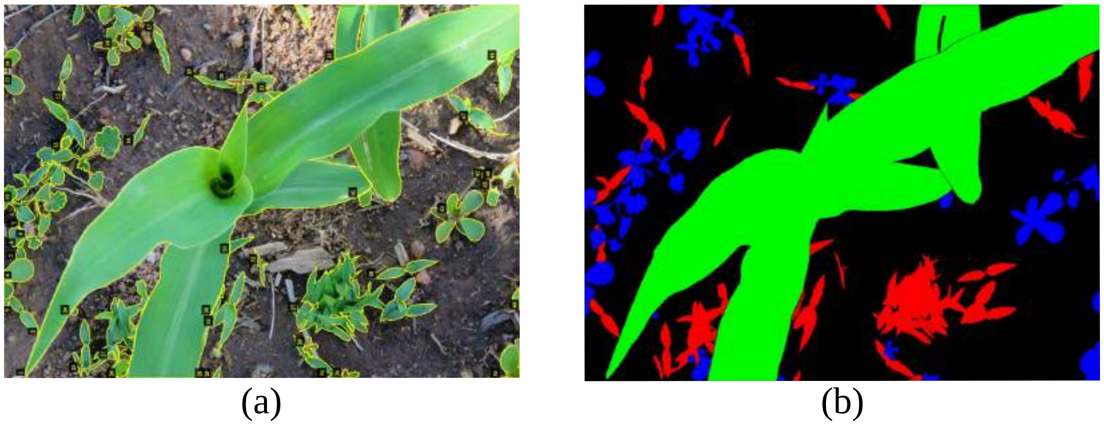


Figure 3.6: Visualization of annotation polygons and image mask with labeled pixels. (a) annotation polygons around the contour of the plants and (b) its corresponding mask in which colors were provided to the regions.

3.1.1.2. Annotation of multispectral images

To annotate the multispectral images, we leverage the visual bands of the camera (RGB). Visual images were obtained by concatenating the RGB bands to distinguish the plants in the frames. Then, the annotation was done in two stages. First, a coarse polygon was also traced in the tool VGG Image Annotator (Dutta and Zisserman, 2019), as observed in Figure 3.8(a). A single polygon could enclose multiple plant species of the same family: corn plants, narrow-leaf weeds and broad-leaf weeds. This annotation includes additional species to that mentioned above. Also, the polygons enclosed background pixels, concerning soil, stones and straws of past crops.

Subsequently, digital image analysis strategies were performed on the coarse annotation images. For this, the annotated image was first split into three

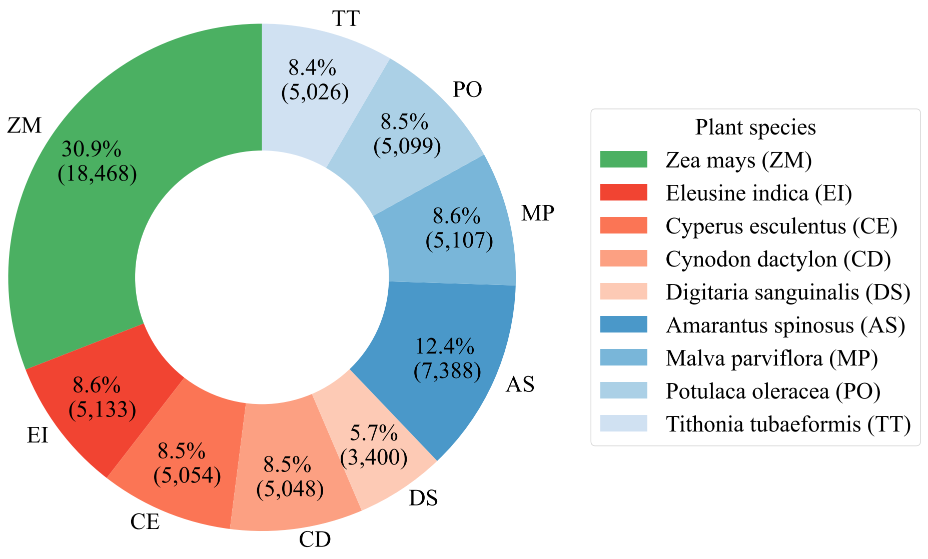


Figure 3.7: The dataset distribution based on plant species considered in this work. The count, percentage, and respective names of annotated instances are observed.

sub-images of the same size as the original; one with all the coarse regions that contain corn plants, another containing the regions of narrow-leaf weeds and the last containing brad-leaf weed regions. For each of these sub-images, the background pixels were eliminated to leave just green foliage belonging to plants.

Background elimination

To explain the process of background elimination, let us define the sub-image $I \in \mathcal{M}^{m \times n \times p}$ as a supermatrix with dimensions $m \times n \times p$, where the ijk-th entry represents the ij-th color pixel for channel k, and $\mathcal{M}^{m \times n \times p}$ represent all hypermatrices of this kind. Although the image is initially in the RGB color space, it has been noted that RGB might not be the most effective choice for distinguishing vegetation from soil (Cheng et al., 2001). Consequently, a color space transformation from RGB to HSV was implemented.

Segmentation in this particular color space was noted to be effective due to the lack of correlation between color (hue channel) and brightness (value channel). This lack of correlation is advantageous for identifying greenness (Yang et al.,

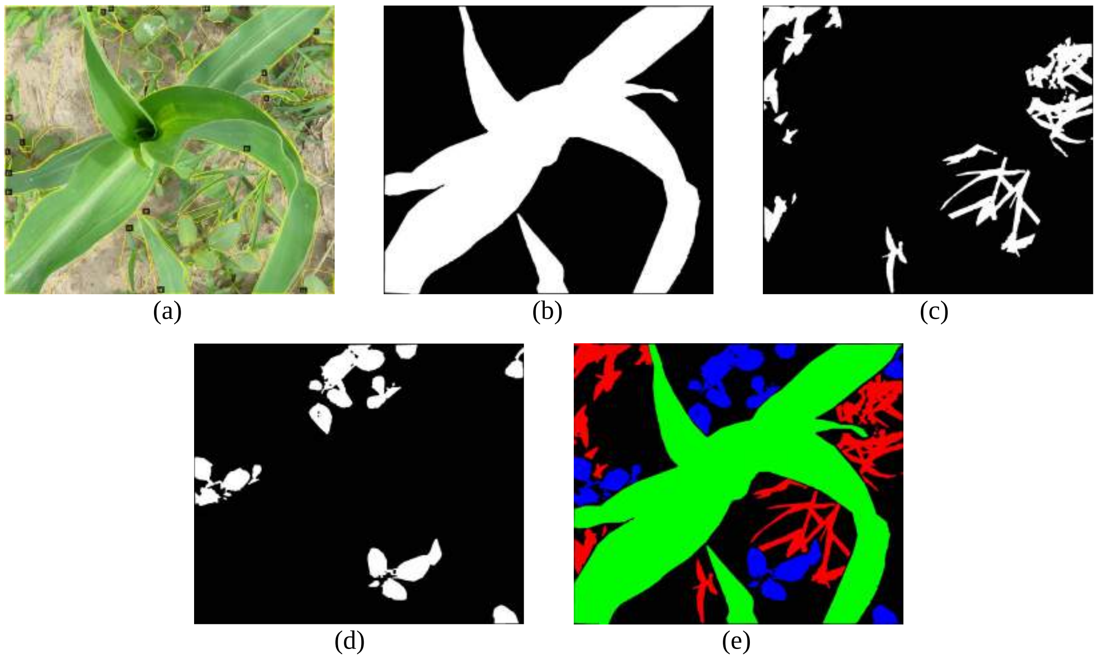


Figure 3.8: Coarse annotation of plants. (a) Polygons enclosing the classes Crop, NLW, and BLW, (b) binary mask of with pixels of the class Crop, (c) binary mask of with pixels of the class NLW, (d) binary mask of with pixels of the class BLW, and (e) labeled image at the pixel level: red regions correspond to the class Crop, red and blue regions correspond to the classes NLW and BLW, respectively.

2015). The resultant image, $I_{hsv} \in \mathcal{M}^{m\times n\times p}$, is employed for background removal, achieved through a thresholding function denoted as, $B : \mathcal{M}^{m\times n\times p} \to \mathcal{M}^{m\times n}$, defined by Equation (3.1).

$$B(x,y) = \begin{cases} 255 & [H_l, S_l, V_l] - I_{hsv}(x,y) \leq 0 \text{ and } I_{hsv}(x,y) - [H_h, S_h, V_h] \leq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $B(x,y) \in \mathcal{M}^{m\times n}$ is the resulting binary image; $I_{hsv}(x,y) = [I_{hsv}(x,y)_h, I_{hsv}(x,y)_s, I_{hsv}(x,y)_v]$ is the vector formed by hue, saturation and value channels of the I_{hsv} sub-image; $H_l, S_l, V_l \in \mathbb{Z}^+$ and $H_h, S_h, V_h \in \mathbb{Z}^+$ are, respectively, the lower and higher values for each of the hue, saturation, and value channels. The threshold values were adjusted manually, and after numerous iterations on images taken under various lighting conditions and natural background variations, the final threshold values were established as follows: $H_l = 33$, $H_h = 95$, $S_l = 34$, $S_h = 255$, $V_l = 60$ and $V_h = 250$. However, the resultant image from this stage contained noise scattered throughout, needing image enhancement.

Image Enhancement

The binary images acquired in the previous stage exhibited numerous gaps within the white areas, which indicate vegetation regions. Also, numerous small regions emerged in areas where theoretically no vegetation was present, signifying the presence of noise. Therefore, the morphological operators opening and closing were executed in the same order to enhance these images. The opening operation serves to refine the contours of images and remove minor artifacts, while the closing operator assists in eliminating small holes and filling gaps in the regions (Le et al., 2020b). But opening and closing are defined by erosion (Equation 3.2) and dilation (Equation 3.3) morphological operators (González and Woods, 2018),

$$A \ominus B = \{z|(B)_z \subseteq A \neq \varnothing\} \tag{3.2}$$

$$A \oplus B = \{z|(\hat{B})_z \cap A\} \tag{3.3}$$

In the erosion operation, A represents all the objects in the binary image, and B is the structuring element. Thus, the erosion of A by B is the set of all points z, such that B translated by z with respect to the origin of B is contained in A. This implies that all overlapping pixels of A and B are replaced by pixels of value 0. Conversely, applying dilation to the binary image A using the structuring element B involves setting pixels to a value of 1 when the center of B aligns with the boundary of A. In this way, opening (Equation 3.4) comprises an erosion operation followed by the dilation operation. Closing (Equation 3.5) operation is defined as a dilation operation followed by an erosion operation.

$$A \circ B = (A \ominus B) \oplus B \tag{3.4}$$

$$A \bullet B = (A \oplus B) \ominus B \tag{3.5}$$

In this thesis a structuring element B of size 5×5 for both opening and closing operations has been used. Subsequently, the connected component analysis (CCA) (Haralick and Shapiro, 1992) was used to eliminate those white regions from the binary image that were of less than 400 pixels. Figure 3.8(b), (c) and (d), respectively, show the resulting binary image of corn plants, narrow-leaf weeds and broad-leaf weeds after applying these morphological operations. Finally, the white regions in the sub-images were assigned a particular pixel vale for corn plants, NLW and BLW. Figure 3.8(e) is the final concatenated annotated image at the pixel level under this procedure.

In total, 2,312 multispectral images were annotated throughout this procedure. The Figure 3.9 indicates the total instances and their corresponding percentage of plants annotated for the classes Crop plants, narrow-leaf weeds and broad-leaf weeds. Therefore, we have the same number of instances of each class for the bans NIR and Red-Edge. As aforementioned, these global classes could include the plant species previously reported and additional ones that have not been identified in this work.

Figure 3.10 provides an overview of the total number of instances in the global classes Crop, BLW, and NLW that constitute our dataset. This count is the sum of instances of the plant species reported above (Figure 3.7) and the multispectral instances with unidentified plant species (Figure 3.9). Therefore, this dataset is composed of 12,887 images.

3.1.2. Crop/weed classification based on shallow and DL

It was proposed the classification of the known plant species of our dataset to assess the performance of a shallow learning approach utilizing LBP+SVM and CNN, to know the route the research should follow. The proposed classification process considers five steps, as shown in Figure 3.11. First, images of the field under natural conditions are acquired. Subsequently, these images undergo background elimination and enhancement using classic image processing techniques (techniques described in Section 3.1.1.2) before advancing to the second stage. In the second stage, regions of interest (ROI) are extracted from the segmented

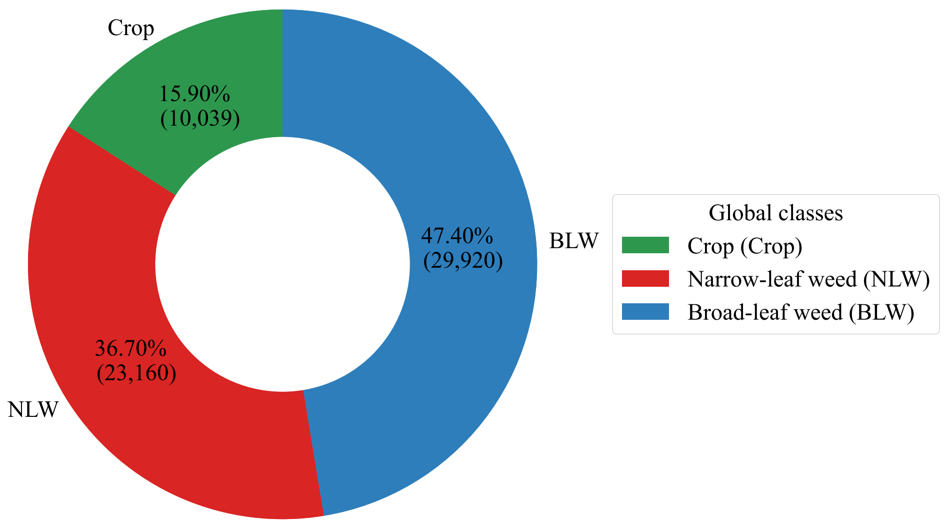


Figure 3.9: The multispectral dataset distribution enclosing multiple plant species. The count and percentage of the annotated images are provided. The given numbers correspond equally for RGB, NIR and RedEdge images captured with the multispectral camera.

image through Connected Component Analysis (CCA) (Haralick et al., 1973; Haralick and Shapiro, 1992). Subsequently, object classification is performed using both LBP+SVM and CNN. For the implementation of classical ML algorithms, the initial step involves extracting texture features through the $LBP_{P,R}^{riu2}$ operator, which has been described in Section 2.2.1. These extracted features are then employed to train a SVM model. The suggested CNN models are derived from established architectures, including VGG16, VGG19 (Simonyan and Zisserman, 2015), and Xception (Chollet, 2017), all of which were trained using our dataset. In the final stage, the vision system reveals the respective class to which each of these objects (plants) is assigned.

The evaluation focused on their effectiveness in classifying plant species, using the carefully annotated dataset described in Figure 3.7. Therefore, the plant species were extracted from the multi-plant images using CCA and manually classified into the classes Crop, NLW, and BLW to construct the experimental dataset for training the models. Figure 3.12 shows a summary of the experimental dataset,

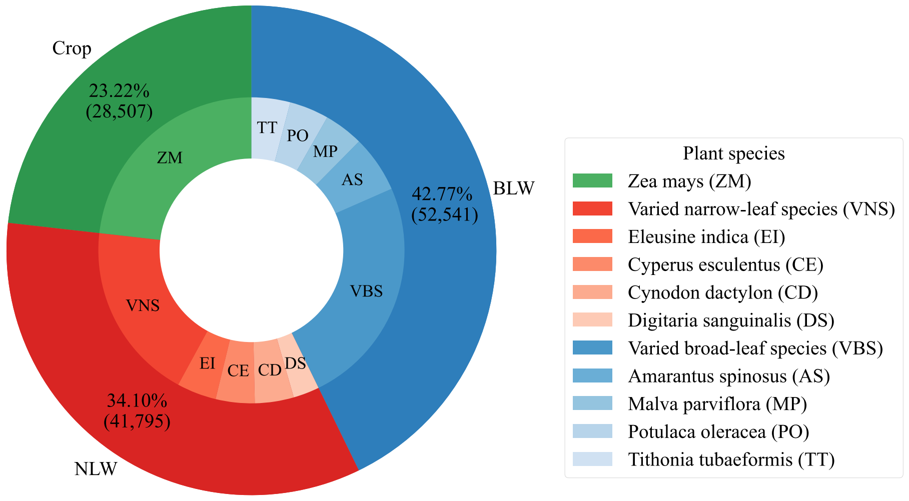


Figure 3.10: The total number of instances annotated in the classes Crop, NLW and BLW in the visible spectrum. This dataset encloses the plant species carefully annotated and the multiple species captured with the multispectral camera. Notice that the main classes NLW and BLW enclose a group with varied narrow-leaf species and another with varied broad-leaf species of weeds.

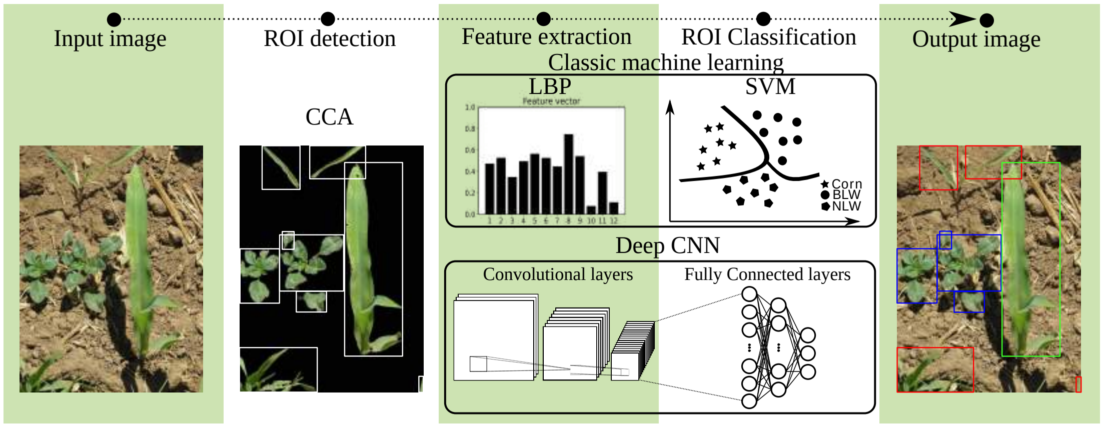


Figure 3.11: A broad overview of the methodology employed for the classification of weeds in real corn fields. Input images undergo background elimination and enhancement using classic image processing techniques. Then the ROIs are extracted through CCA. In the third stage the ROIs are classified with both LBP+SVM and CNNs. In the output image, green box is Crop class, red boxes are the class NLW and blue boxes are the class BLW.

in which the number of instances that integrate each class is provided.

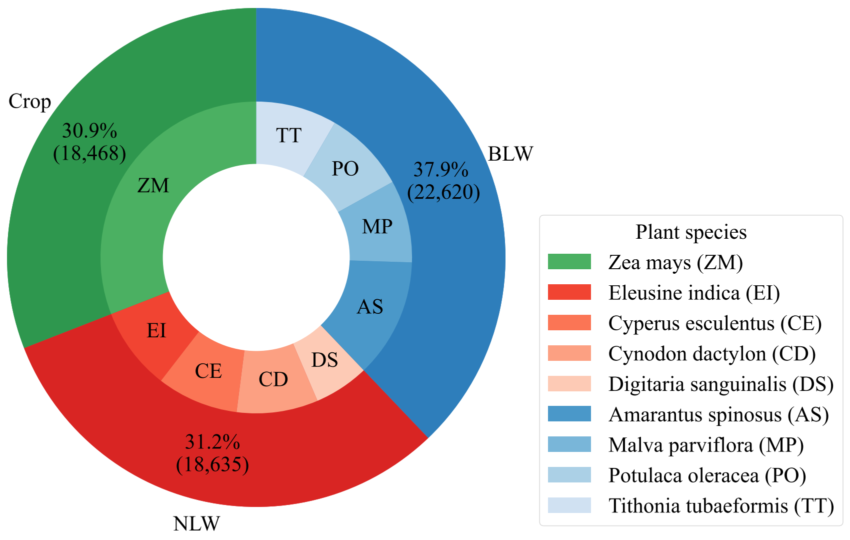


Figure 3.12: Experimental dataset grouped into the classes Crop, NLW and BLW.

3.1.2.1. Classical machine learning approach

The proposed classical approach consists of three stages, as shown in Figure 3.13. In the initial stage, the RGB image is obtained and undergoes preprocessing, involving a color space transformation from RGB to grayscale. Subsequently, in the second and third stages, texture feature extraction and classification are performed, respectively. The rotation invariant local binary pattern ($LBP_{P,R}^{riu2}$) was employed as texture descriptor, covered in Section 2.2.1. Whereas for the classification of the descriptors, SVM models were used, which was mathematically explained in Section 2.2.2.

3.1.2.2. Classification based on CNNs

In Section 2.3, the operational principles of ANNs and CNNs were discussed. The architecture of CNNs comprises convolutional layers and fully connected

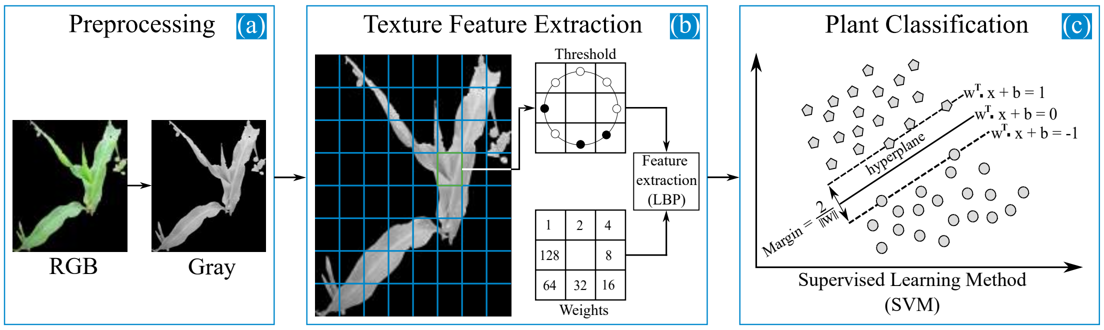


Figure 3.13: Overview of the classification methodology utilizing traditional machine learning. (a) Transformation of the input image into a different color space, (b) extraction of texture features from the input image using $LBP_{P,R}^{riu2}$, and (c) classification of the extracted texture features through SVM.

layers. Convolutional layers function to extract and learn features from images, while the fully connected layers, essentially an ANN, are responsible for classifying the features extracted by the convolutional layers.

Under this classification approach, it was proposed the networks VGG16, VGG19 (Simonyan and Zisserman, 2015) and Xception (Chollet, 2017) because they have demonstrated outstanding efficacy in tasks related to plant classification (Ahmad et al., 2021; Espejo-Garcia et al., 2020; Le et al., 2020a). Another justification for utilizing the VGG networks is its ability to deliver strong performance in terms of accuracy, even when trained on a dataset containing a limited number of images (Theckedath and Sedamkar, 2020). Therefore, below are described the main parameters of VGG16 and Xception.

VGG networks

The convolutional layers of the VGG architectures, also called Visual Geometry Group, are clustered in blocks. Each of these blocks could perform two or three consecutive convolutional operations and a ReLu normalization, followed by a max-pooling operation, as shown in Figure 3.14. The convolutional operations use kernels of size 3×3. This kernel size is smaller compared to those implemented in other CNNs proposed before the epoch the networks were launched, which usually use kernels of size 5×5, 7×7 and 11×11. The kernel follows a stride of 1 and zero padding in the convolutional operations to conserve the spatial resolution of the input feature map. The benefit of using small-size filters lies in their ability to extract features as effectively as larger-size filters. Additionally, employing smaller filters leads to a reduction in the number of parameters, consequently lowering computational costs Alzubaidi et al. (2021). Regarding the max-pooling operations in these networks, they utilize 2×2 size kernels with a stride of 2.

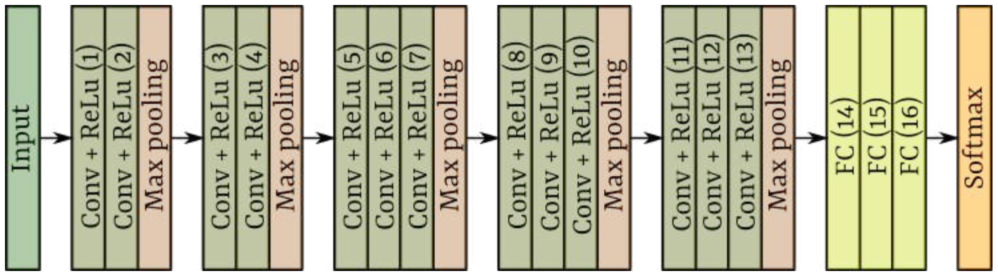


Figure 3.14: Representation of the standard VGG architecture.

The FCL of these networks are composed of three layers. The first two layers have 4096 channels with ReLu activation function. The last layer of the FCL depends on the number of classes to be classified; for this reason, it comes with a softmax activation function. The numbers 16 and 19 in VGG16 and VGG19 refer to the number of layers with learnable parameters. Figure 3.14 shows the VGG16 standard architecture. In the case of VGG19, three more consecutive convolution blocks followed by a max-pooling layer are added.

Xception network

Xception is a CNN that drew inspiration from Depthwise Separable Convolutions (DSConv) and Inception modules (Chollet, 2017).

The initial documented DSConv (Szegedy et al., 2015) is structured with Depthwise Convolutions (DC) preceding Pointwise Convolution (PC). In DC, spatial convolution is performed individually by the filters over each input data channel. Subsequently, the PC transforms the output feature map into another channel dimension while preserving its spatial size through a 1×1 convolution. It is noteworthy that DSConv does not incorporate any activation function between DC and PC.

Regarding the Inception module (Szegedy et al., 2017), the DSConv is employed in the reverse sequence. Initially, PC is applied to the input data, followed by

DC. Additionally, unlike the original DSConv, an inception module introduces an activation function between PC and DC. The concept behind an inception module is to first capture cross-channel correlations through 1×1 convolutions and then condense these correlations into a smaller channel dimension. Thus, a typical inception module performs three 1×1 convolutional transformations (PC) and a max-pooling operation concurrently. These are followed by 3×3 and 5×5 convolutions (DC). The outcomes of these operations are then consolidated into a single feature map, maintaining the dimensions of the channels.

Regarding an Xception module, alike the Inception module, it initiates by performing PC to capture cross-channel correlations, followed by mapping the spatial correlation of each output channel through DC. However, the Xception module incorporates a single PC. To illustrate this concept more clearly, a module of the Xception network is depicted in Figure 3.15. Similar to the original DSConv, Xception does not include any activation functions between PC and DC. The core idea of Xception is to reduce computational cost and maintain the number of parameters, like the approach in Inception.

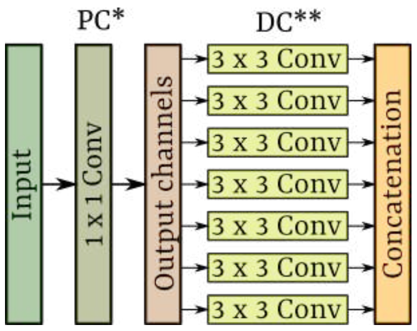


Figure 3.15: Representation of Xception module. *Pointwise convolution; **Depthwise convolution.

3.1.2.3. Experimental setup for shallow and DL classification

Classical machine learning approach

A series of experiments were conducted to assess the effectiveness of the suggested traditional ML method in the task of classification. As previously stated, texture features were extracted using the $LBP_{P,R}^{riu2}$ operator, and the SVM was employed for the classification of the resulting vector features. Regarding the LBP operator, three distinct combinations of spatial and angular resolutions (P, R) were employed, specifically with values of $(8, 1)$, $(16, 2)$, and $(24, 3)$. In addition, three different image sizes have also been tested, 256×256, 128×128, and 64×64 pixels, and depending on this size, they were also divided into cells of size 8×8, 16×16, 32×32, 64×64 and 128×128, as shown in stage two of Figure 3.13. The following Table 3.1 lists the set of treatments.

Table 3.1: Arrangement of the experimental dataset for the extraction of texture features.

		Cell Size				
$\boldsymbol{LBP^{riu2}_{P,R}}$	**Image Size**	**8 × 8**	**16 × 16**	**32 × 32**	**64 × 64**	**128 × 128**
P = 8, R = 1	256 × 256	✓	✓	✓	✓	✓
	128 × 128	✓	✓	✓	✓	
	64 × 64	✓	✓	✓		
P = 16, R = 2	256 × 256	✓	✓	✓	✓	✓
	128 × 128	✓	✓	✓	✓	
	64 × 64	✓	✓	✓		
P = 24, R = 3	256 × 256	✓	✓	✓	✓	✓
	128 × 128	✓	✓	✓	✓	
	64 × 64	✓	✓	✓		

The $LBP^{riu2}_{P,R}$ operator generate feature vectors of size "$P + 2$". That is, the operators $LBP^{riu2}_{8,1}$, $LBP^{riu2}_{16,2}$, and $LBP^{riu2}_{24,3}$ generate output vectors containing 10, 18, and 26 elements, respectively. As a result, the overall length of the concatenated feature vector for each configuration was determined by both the image size and the number of cells in the images

SVM classifiers underwent previous parameter tuning. I was determined that the most suitable kernel function for our data, based on accuracy, was linear, indicating that weights were not transformed. The configuration for the C value, which sets an upper limit on the Lagrangian optimization variables, was tuned by initiating in 1 and gradually increasing by one unit. The optimal accuracy in tuning was achieved when $C = 5$. Concerning the dataset, it was partitioned into 70 % for training, 20 % for validation, and 10 % for testing. The implementation was performed using Python 3.8. The training process took place on a laptop computer equipped with an Intel Core i7-8550U processor, Intel UHD Graphics 620, and 16 GB of RAM.

Classification based on CNNs

For each of the three CNN architectures, the convolutional layers were preserved and their FCL were replaced for our proposal. In this regard, the configuration of the FCL for each model was of two layers. The input layer of 512 channels, followed by a ReLu activation function. The output layer consists of three neurons, the same number as the classes of our dataset, followed by the softmax activation function.

The training procedure took place on a desktop computer featuring a Core i7 10700 processor, an NVIDIA Quadro P400 graphics processing unit (GPU), and 8 GB of RAM. The implementation was carried out using Python 3.8 and the Keras framework with a Tensorflow 2.5.0 backend. The experimental dataset was divided into 70 % for training, 20 % for validation, and 10 % for testing. Additionally, the images were resized to $128 \times 128 \times 3$ pixels for all three models.

The transfer learning strategy was applied to the convolutional layers, preserving and retraining the weights initially tuned in the ImageNet dataset. A dropout regularization of 0.5 was implemented in the fully connected layers. Given our dataset's three classes, training utilized the *categorical crossentropy* loss function, and the Adam optimizer was employed with a learning rate of 0.0001. All models underwent training for 100 epochs with a batch size of 16.

Evaluation of performance

The performance of the two approaches in classifying the plant species grouped into the classes Crop, NLW and BLW was implemented with the metrics accuracy, precision, recall and F1 score, whose mathematical expressions were given in Section 2.4.

3.1.3. Segmentation based on R-CNN and U-Net-like

The detection of crop plants, NLW and BLW within natural corn fields could be addressed through supervised pixel-wise semantic segmentation. That is to say, given an image acquired from a natural cornfield, denoted as $I^{m\times n\times 3}$ where $m \times n$ refers to the spatial size of the image. Here, the problem is to classify each pixel ($[x_i]_{r,c}$, $r = 1, \ldots, m$; $c = 1, \ldots, n$) into the classes Crop, NLW, BLW, or Soil. Therefore, this section covers the labeling of each pixel of input images based on deep-learning models.

Figure 3.16 shows the strategy that was followed to address the problem of pixel-wise semantic segmentation. In this process, the input images may contain multiple plants, typical of an authentic field scenario, which have to be segmented by deep CNNs. The studied segmentation CNNs were the well-known Mask R-CNN and a proposed improvement of this network that we called Mask R-CNN-ASPP and finally a U-Net-like architecture.

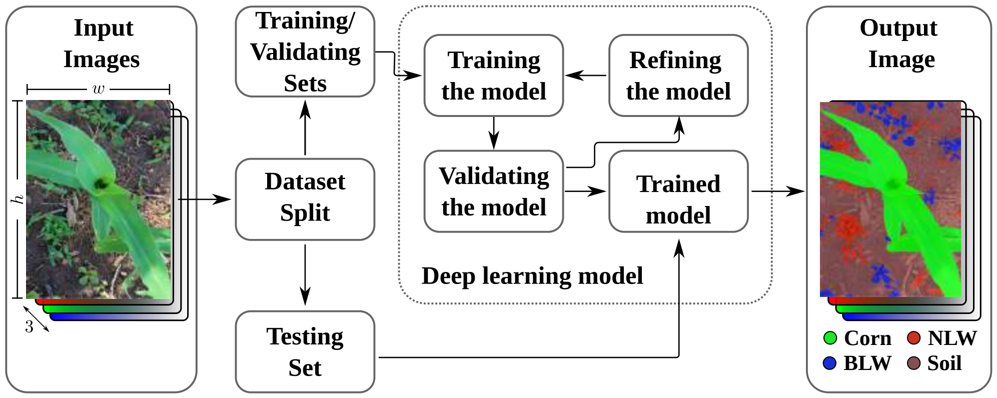


Figure 3.16: Overview of the approach for semantic segmentation of crops and weeds in natural corn fields. The image captured at ground level is displayed on the left. In the center, the series of steps for training, validating, and testing the deep learning model is depicted. On the right, the resulting segmented image is presented. The segmented crop plant is highlighted in green, NLW are shown in red, and BLW are represented in blue.

The dataset used for training the three models was that summarized in Figure 3.12. This dataset encloses all the plant species that were carefully annotated and captured using RGB cameras. The plant species in this dataset have been grouped into the classes Crop, NLW and BLW.

3.1.3.1. Description of the convolutional networks

In this section, a brief explanation of the architectures Mask R-CNN, the proposed Mask R-CNN-ASPP and the U-Net-like architecture is presented.

Mask R-CNN

The Mask R-CNN is specialized for object detection and instance segmentation He et al. (2017). Figure 3.17 depicts the Mask R-CNN model, where the backbone is responsible for taking the input image and generating a feature map. This feature map undergoes analysis by a Region Proposal Network (RPN), producing rectangular region proposals. However, these proposed regions are misaligned concerning the input image. Consequently, a ROI alignment process correctly aligns these ROIs with respect to the input image. All these components collectively form the mapping (f_θ) of the input image into a fixed-size feature map.

On the other hand, the head of the architecture consists of two parallel branches. The first branch is a fully connected layer, f_ϕ for predicting a bounding box for each ROI and further classification. The second branch is a FCN, f_γ, for predicting a binary mask for each class, which is independent of the classification branch. The FCN comprises four consecutive 3×3 convolutional layers, followed by a 2×2 deconvolutional layer with a stride of 2, and finally, a 1×1 convolutional layer. All these hidden layers use the ReLu activation function. Consequently, instance segmentation is performed over the objects. The overall process can be summarized as follows: from each input image x_i, a feature map $\mathcal{F} = f_\theta(x_i)$ is computed, serving as the input for both a fully connected layer for classification, $f_\phi(f_\theta(x_i))$, and an FCN for instance segmentation $f_\gamma(f_\theta(x_i))$.

Mask R-CNN-ASPP

The Mask R-CNN network utilizes a segmentation branch composed solely of convolutions and deconvolutions. However, this approach has limitations, as convolutions without additional operations fail to extract essential spatial context information from the feature maps. This lack of spatial context information becomes particularly critical for improving segmentation, especially when dealing with a high density of objects to be segmented, as observed in our study. To

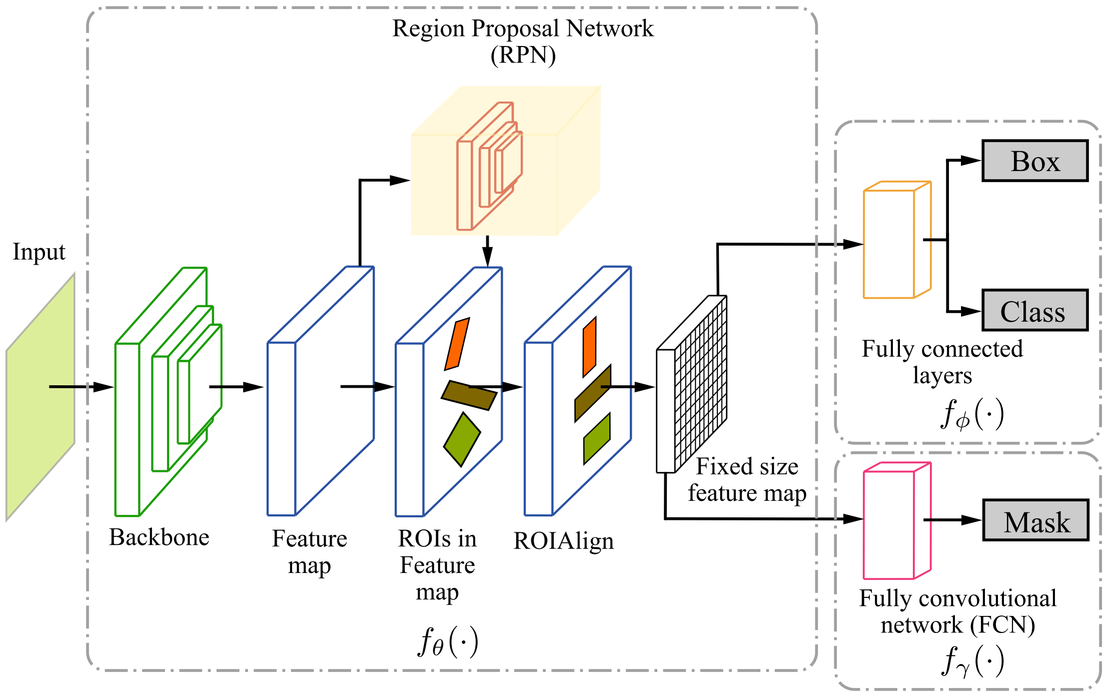


Figure 3.17: Representation of the Mask R-CNN used for semantic segmentation of herbs and corn plants.

address this limitation, we propose a strategy for enhancing the segmentation of corn and weed plants by implementing Atrous Spatial Pyramid Pooling (ASPP) within the FCN branch of the Mask R-CNN architecture. The ASPP module utilizes *atrous convolutions*, also known as dilated convolutions, which involve convolutions incorporating pixels positioned at a specified distance from the central pixel rather than solely using adjacent pixels. This distance is determined by the dilatation rate (r). Through the utilization of *atrous convolutions*, the ASPP module facilitates the enlargement of the filter's field of view (Chen et al., 2017). In this thesis, we integrate the ASPP module into the Mask R-CNN architecture, as depicted in Figure 3.18.

The ASPP takes each fixed-sized feature map (ROI) calculated by the ROIAlign block as input. Subsequently, the ASPP block applies three dilated convolutions and a pooling operation to each input ROI. The dilated convolutions utilize dilatation rates of one ($r = 1$), three ($r = 3$), and six ($r = 6$). Following the *atrous convolutions*, batch normalization is performed, succeeded by a ReLu activation

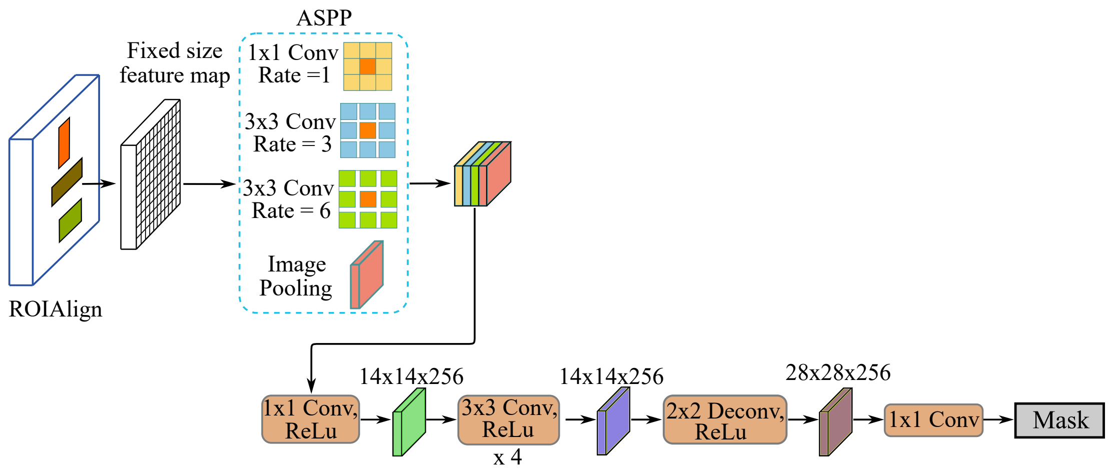


Figure 3.18: The ASPP module joined the segmentation branch of the Mask R-CNN to improve the segmentation of corn and weed plants.

function. Subsequently, an image pooling operation, specifically a 2×2 average pooling, is applied to each input ROI, followed by upsampling by a factor of 2 using bilinear interpolation. The four resulting outputs are concatenated and then convolved with a standard 1×1 kernel, followed by a ReLU activation function. This process yields a feature map of dimensions $14 \times 14 \times 256$. The subsequent operations are in line with the original FCN of the Mask-RCNN architecture.

U-Net-like architecture

The U-Net-like architecture comprises two primary components: an encoder, also known as the backbone or contracting path, and a decoder, which is the expansive path. The encoder executes convolutional operations to extract crucial features. Conversely, the decoder utilizes transposed 2D convolutional layers to enlarge the feature blocks until they align with the dimensions of the original input image. In our implementation, we adopt the ResNet50 and ResNet101 architectures (He et al., 2016) to function as the encoder component of our model. The U-Net-like architecture is visually depicted in Figure 3.19.

Starting from the input image, the encoder operations initiate with a 7×7 padded convolution, followed by normalization and the application of a ReLu activation function. These consecutive operations generate an initial feature map with dimensions of $256 \times 256 \times 64$. Following this, the said feature map serves as the input for the "ResNet, B1" block, and the output from this block is subsequently forwarded to the next "ResNet, B2" block. This sequence persists until the final output is derived from the "ResNet, B4" block. Each ResNet block reduces the spatial dimension and increase the deep of the feature maps, resulting in halved dimensions and twice the number of channels in comparison to the preceding stage, as illustrated in Figure 3.19.

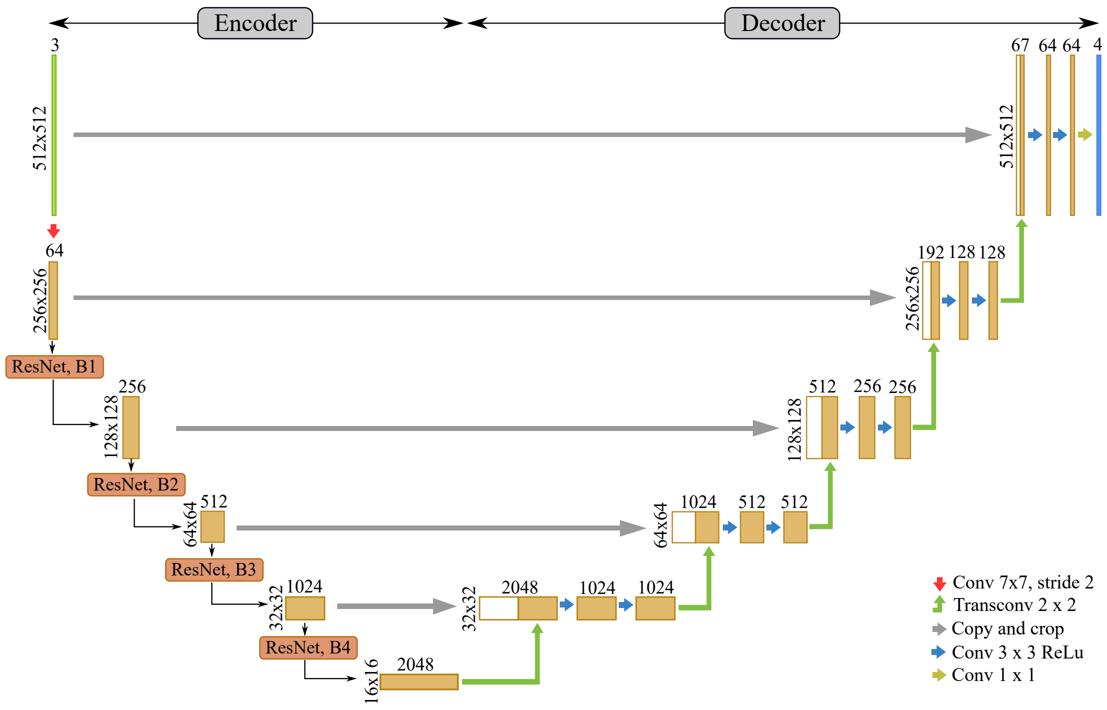


Figure 3.19: The U-Net-like architecture representation for semantic segmentation of weed plants and corn crops. ResNet50 and ResNet101 convolutional layers function as the encoder of the models. Each ResNet block reduces the spatial dimension and increase the deep of the feature maps.

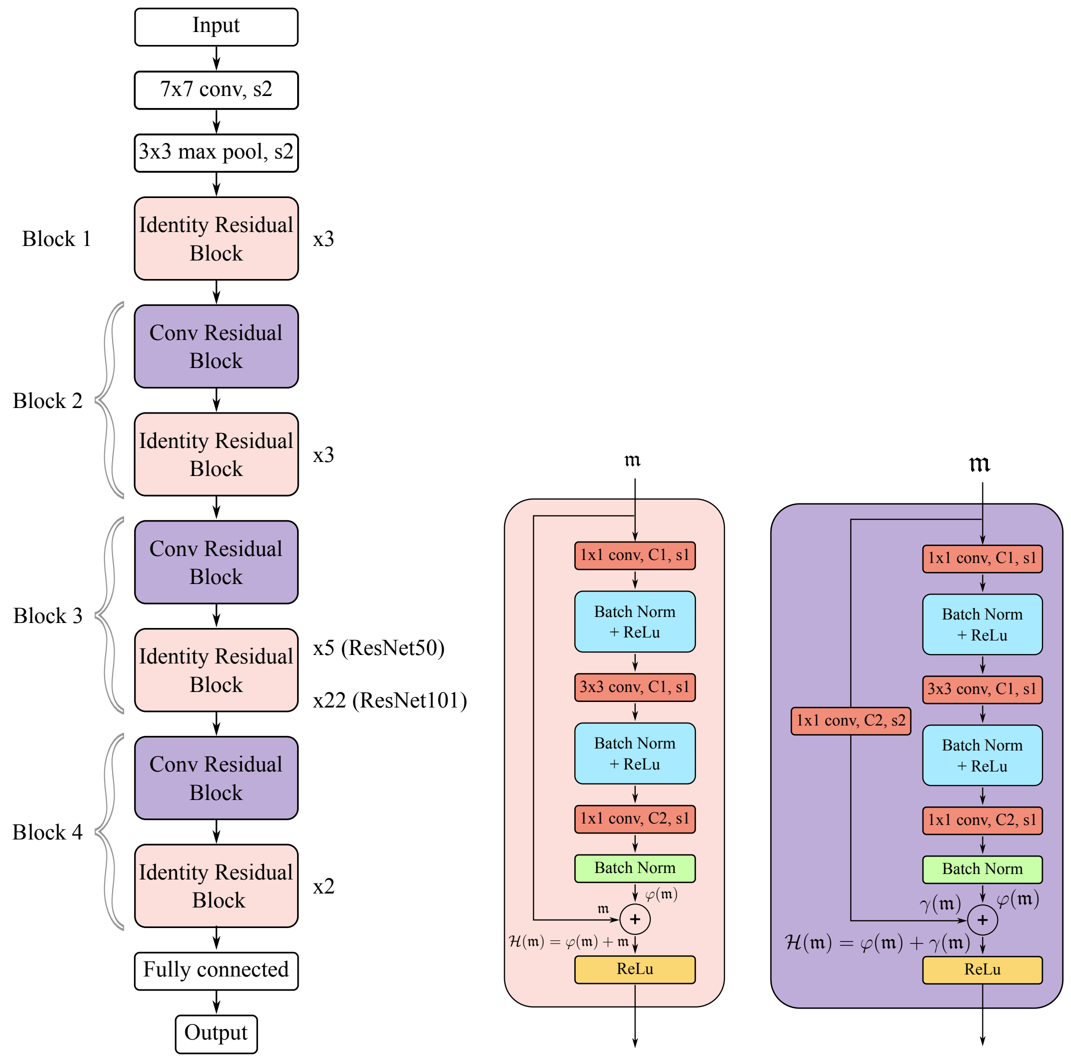
In the decoder segment of our proposed network, we utilize 2×2 transposed convolutions to facilitate the up-sampling of feature maps at each stage. This operation effectively doubles the size of the feature maps while reducing the number of channels by half. Subsequently, the up-sampled feature maps are concatenated with the corresponding feature map obtained from the ResNet block at the same level in the encoder. Following the concatenation, two 3×3 padded convolutions and the ReLu activation function are applied. Finally, at the concluding layer of the decoder, a 1×1 convolution is employed to map each 64-dimensional feature vector to a four-channel output. This number of output channels corresponds to the classes present in our dataset.

3.1.3.2. Details of the ResNet backbone

The U-Net-like architecture is built upon the incorporation of ResNet50 and ResNet101 models, chosen for their demonstrated effectiveness in plant classification in natural environments, as evidenced by prior studies (Peng et al., 2022; Picon et al., 2022; Quan et al., 2021; Zenkl et al., 2022). Both the ResNet50 and the ResNet101 architectures involve a 7×7 padded convolution layer with a stride of 2, followed by a 3×3 max pooling layer with the same stride. This is succeeded by four consecutive main blocks, each housing residual blocks with peculiar properties. These main blocks are linked to a fully connected layer, which, in turn, connects to the output layer responsible for generating the final predictions. Figure 3.20 provides an overview of the entire architecture, illustrating the arrangement of the residual blocks.

The ResNet architecture is distinguished by the incorporation of residual building blocks, utilizing two types: the *identity block* (illustrated in Figure 3.20(b)) and the *convolutional block* (shown in Figure 3.20(c)). The *identity block* is applied when the input feature map ($\mathfrak{m}$) and the output feature map of the block ($\varphi(\mathfrak{m})$) share identical dimensions.

As depicted in Figure 3.20(b), the *identity block* comprises three consecutive convolutions (1×1, 3×3, and 1×1), each followed by a normalization operation and a $ReLU$ activation function. The resulting output is then element-wise added to the feature map ($\mathfrak{m}$) and directed into the residual block via a shortcut path. This addition results in the output $\mathcal{H}(\mathfrak{m})$, representing the underlying mapping. Notably, the number of kernels utilized in the *identity block*, denoted as “C1” and “C2” varies based on the specific main block (Block 1, Block 2, Block 3, or Block 4) within the ResNet architecture. For example, in the first main block (Block 1), $C1 = 64$ and $C2 = 256$, while in the second main block (Block 2), $C1 = 128$ and $C2 = 512$, and so forth. This variation allows the network to capture diverse levels of complexity and abstraction.



(a) ResNet architecture (b) Identity residual block (c) Convolutional residual block

Figure 3.20: The ResNet architecture, which was adopted as the backbone for the proposed U-Net-like architecture. (a) Configuration of the main blocks of both the ResNet50 and the ResNet101. (b) Inside structure of the identity residual block applied in both ResNet50 and ResNet101. This block is employed when the dimensions of the feature maps remain unchanged. (c) The inside structure of a convolutional residual block employed for transition steps. This block is used when there exists a reduction in the size of the feature maps.

In contrast to traditional CNNs that stack convolutional layers to approximate the input, the use of residual blocks offers an advantage by having the network learn the residual map, denoted as $\varphi(\mathfrak{m}) = \mathcal{H}(\mathfrak{m}) - \mathfrak{m}$. This formulation helps address the vanishing gradient problem, as when $\varphi(\mathfrak{m})$ tends toward zero during backpropagation, the identity map $\mathfrak{m}$ contributes to non-zero weights. Consequently, gradients are propagated to the initial layers of the network, enabling them to adjust their parameters, which facilitates the training of deeper networks.

In cases where the input and output possess different dimensions, the *convolutional block* is employed. Unlike the identity block, the convolutional block includes a 1×1 convolutional layer in the shortcut path, along with a variation in the number of kernels. Specifically, for the convolutional block, the values of $(\mathrm{C1}, \mathrm{C2})$ are selected from the set$\{(128, 512), (256, 1024), (512, 2048)\}$. It is noteworthy that the convolutional block is not present in the first main block (Block 1). The inclusion of the 1×1 convolutional layer in the shortcut path allows for adjusting the dimensions of the feature maps to align with the desired output size. This additional convolutional layer aids in incorporating richer spatial information and adapting the network's capability to accommodate variations in spatial resolution throughout the network. Nevertheless, in the first main block (Block 1), where the initial feature maps are obtained, the convolutional block is unnecessary since the dimensions of the input and output feature maps are the same.

A notable difference between ResNet50 and ResNet101 lies in the quantity of residual blocks within the main Block 3. Particularly, ResNet50 includes five residual blocks, whereas ResNet101 incorporates twenty-two residual blocks. taking into account the common 7×7 convolutional and 3×3 max pooling layers in both networks, ResNet50 has a total of 50 layers, while ResNet101 includes a total of 101 layers. The variation in the number of residual blocks between the two architectures significantly influences their depth and capacity to capture complex patterns and features in the input data. With a greater number of layers and residual blocks, ResNet101 possesses a more extensive and expressive network structure, enhancing its ability to represent increasingly complex relationships and learn hierarchical features. However, it is important to note that the deeper architecture of ResNet101 may pose challenges, including increased computational requirements and a potential risk of overfitting, especially in scenarios with limited training data. As a result, the decision between ResNet50 and ResNet101 depends on the specific requirements of the task, requiring a balance between model complexity and computational efficiency.

3.1.3.3. Experimental setup for segmenting crop and weeds based on CNNs and RGB images

The following six experiments for pixel-wise semantic segmentation using our dataset (Figure 3.12) in the visible spectrum were carried out:

- Mask R-CNN-ResNet50
- Mask R-CNN-ResNet101
- Mask R-CNN-ASPP-ResNet50
- Mask R-CNN-ASPP-ResNet101
- U-Net-like-ResNet50
- U-Net-like-ResNet101

ResNet50 and ResNet101 (He et al., 2016) have been used as backbones for Mask R-CNN, Mask R-CNN-ASPP and U-Net-like. Furthermore, transfer learning was employed; therefore, the resulting weights after having trained the ResNet50 and ResNet101 networks in the well-known ImageNet dataset (Deng et al., 2009) were used to start the training and then tuned to our dataset. Moreover, to ensure equal representation of instances per plant class, the dataset was balanced, equalizing the instances into the classes Crop, NLW and BLW in $22,620$ instances per class to avoid overfitting and improve the performance of the models. Subsequently, the dataset was split in into 70 % for training, 20 % for validation and 10 % for testing the models.

It is worth mentioning that to train the models, a desktop computer with a Core i7 processor, NVIDIA GPU GeForce GTX 1080Ti with 6 GB of VRAM, and 64 GB of RAM has been used. The implementation was carried out in Python 3.8 and Keras framework with Tensorflow 2.4.0 as a backend.

Mask R-CNN and Mask R-CNN-ASPP training

Mask R-CNN and Mask R-CNN-ASPP are designed for the instance segmentation of objects. Consequently, we have associated each segmented object with its respective class (corn, NLW, NLB) to provide a clear interpretation of the image. In other words, if an image contains "n" objects, each object is labeled distinctly as either corn, narrow-leaf weeds or broad-leaf weeds.

The two networks have been configured to support an input image ($x_i \in I^{n \times m}$), with maximum dimensions of 1024×1024. Furthermore, since some images from the dataset have around 250 labels, the RPN was configured to train 500 anchor boxes and ROIs per image.

The models were trained for 200 epochs with a batch size of 1. During this process, the weight decay and the learning rate were set to 0.0001, and the stochastic gradient descent (SGD) was used as optimizer.

To adjust network parameters to our dataset, the loss function proposed by He et al. (2017), has been used. This loss function is defined as follows,

$$L = L_{cls} + L_{box} + L_{mask} \tag{3.6}$$

where L represent the total loss function of the model; L_{cls} is the classification loss; L_{box} is the bounding box regression loss; L_{mask} is the average binary cross-entropy loss.

Particularly, the classification loss (L_{cls}) is computed according to,

$$L_{cls} = \frac{1}{N_{cls}} \sum_i -log[p_i p_i^* + (1 - p_i^*)(1 - p_i)] \tag{3.7}$$

where N_{cls} are the number of categories; p_i is the probability that the $i - th$ ROI is predicted to be the target. Here, when the predicted ROI is foreground, $p_i^* = 1$, otherwise, $p_i^* = 0$.

On the other hand, the bounding box regression (L_{box}) loss is computed by the following expression,

$$L_{box} = \frac{1}{N_{box}} \sum_i p_i^* R(t_i, t_i^*) \tag{3.8}$$

where N_{box} is the number of pixels in the feature map; $R(\cdot)$ is a smooth function; t_i represents the four parameterized coordinate vector of the predicted ROIs; and t_i^* indicates the coordinate vector of the real label.

Finally, the computation of the mask loss (L_{mask}) is given by,

$$L_{mask} = -\frac{1}{N}\sum_i [y_i^* log(p(y_i)) - (1-y_i^*)log(1-p(y_i))] \tag{3.9}$$

where N represents the number of pixels; y_i^* is the predicted $k-th$ class in that location of the pixel; and $p(y_i)$ is the probability of the y_i predicted category.

U-Net-like training

For training the U-Net-like model, the input image, $x_i \in I^{n\times m}$, was scaled to a size of 512×512 ($\mathcal{S} : I^{n\times m} \to I_s^{512\times512}$), then, this image was mapped according to $\mathcal{L} : I_s^{512\times512} \to [0,1]^{512\times512} \cap \mathbb{R}^{512\times512}$, i.e., the image was normalized in such a way that each of its elements is in the range $[0,1] \cap \mathbb{R}$.

Also, the model has been trained for 200 epochs with a batch size of one. The SGD optimizer was used with a learning rate of 0.0001.

The dice loss function was implemented to calculate the error between the ground truth image and the predicted mask image. On the other hand, the focal loss function was used to compensate for the complicated finding of the NLW class pixels, since it usually occupies a big area but a low number of pixels in the image, due to the phenological appearance of the plant species. These loss functions were mathematically expressed in Section 2.3.1.3.

Evaluation of performance

The performance of the semantic segmentation in classifying the pixels of plant species grouped into the classes Crop, NLW and BLW was implemented with the metrics DSC, IoU, and mIoU whose mathematical expressions were provided in Section 2.4.

3.1.4. Segmentation and classification for detection

Based on the findings proposed in Section 3.1.3, we realized that the segmentation of plants is effectively carried out by the U-Net-like model, utilizing

ResNet101 as its encoder layer. However, a notable drawback is observed in the misclassification of pixels within isolated ROIs. This leads the thesis to explore other strategies for detecting corn plants (Crop), NLW, and BLW in real corn fields.

To address this gap, we proposed a detection method based on deep learning segmentation and classification networks, as illustrated in Figure 3.21. The algorithm comprises two main stages: segmentation and classification. In the segmentation stage, an image featuring multiple plants is segmented using a U-Net-like-ResNet101 architecture. The segmentation process is approached in two ways. The first involves a simple step of resizing the input images by using the U-Net-like-ResNet101 architecture, while the second step entails dividing the input images into patches to prevent the loss of crucial features, followed by segmenting each patch. To enhance the U-Net-like model's performance in pixel classification for the Crop, NLW, and BLW classes, a series of experiments were conducted using both multispectral and visible spectrum datasets, as explained ahead in Section 3.1.4.2.

Moving to the classification stage, the pixels corresponding to each class (Crop, NLW, or BLW) in the segmented image are initially separated into single-class images. Subsequently, each image undergoes a transformation into binary masks, facilitating the easy extraction of ROIs in scenarios with a high plant density. These ROIs are extracted using the well-known connected component analysis (CCA) (Haralick et al., 1973; Haralick and Shapiro, 1992). For the classification of the ROIs, the networks ResNet101, VGG16, Xception, and MobileNetV2 are implemented and evaluated. Ultimately, the process yields an image in which the plants are successfully detected.

3.1.4.1. Description of classification network

In classification networks, the convolutional layers are arranged in a downsampling configuration to derive a feature map from an input image. Subsequently, a set of FCL is employed to classify the pixels within the feature map. On this regard, earlier in Section 3.1.2 a brief description of the classification networks VGG and Xception was presented and the architecture ResNet101 was

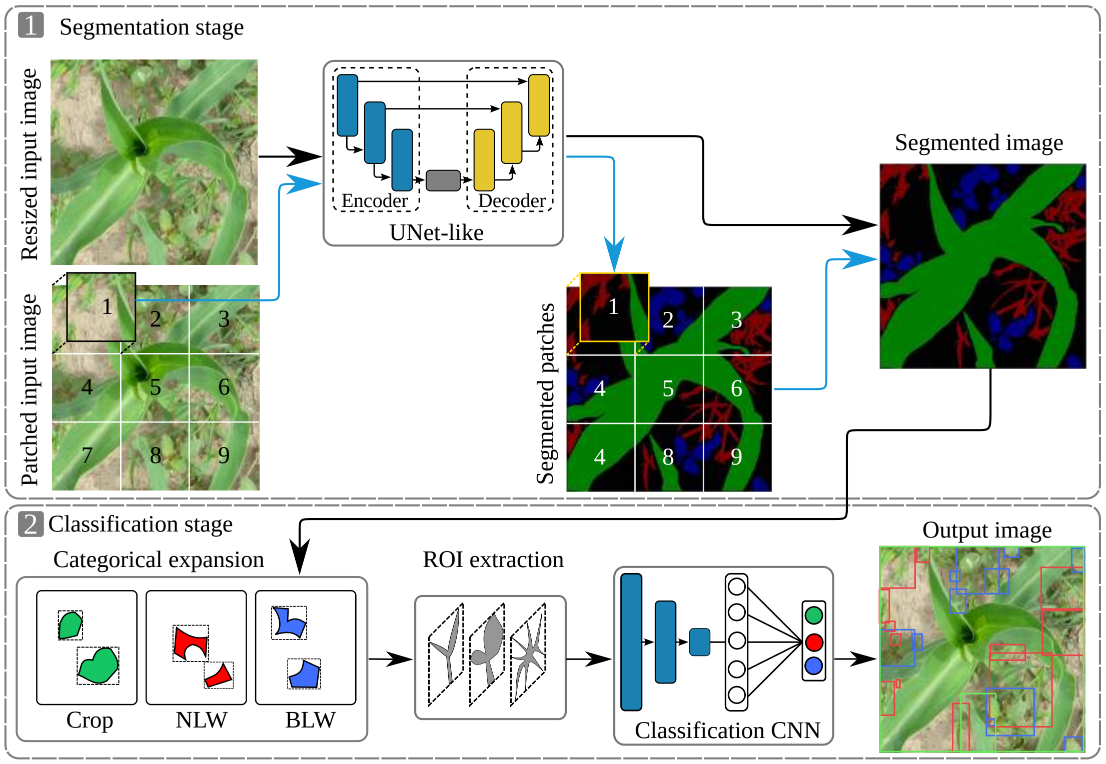


Figure 3.21: The suggested approach for identifying crop and weed plants in real corn fields utilizing segmentation and classification networks. In the resulting image, the classes Crop, NLW, and BLW are represented by the green box, red box, and blue box, respectively.

described in Section 3.1.3.2. Therefore, here a brief explanation of the classification network MobileNetV2 implemented in this approach is presented.

MobileNetV2

The MobileNetV2 architecture (Sandler et al., 2018) is a learning model that uses the common operations of *convolutions*, *activation functions*, and *pooling*. Figure 3.22 depicts the architecture representation of this network. It was designed for three channels images of size 224 × 224 pixels, and its operation blocks start with a common *conv2d*, which is followed by 17 consecutive Bottleneck Depthwise Separable Convolution blocks (BDSC). Then, what follows the last BDSC block is newly a common *conv2d* of filters of size (1 × 1); whose output feature map is applied an *avgpool* operation of a kernel of size (7 × 7); finally, a *conv2d* ends the block operations of the network.

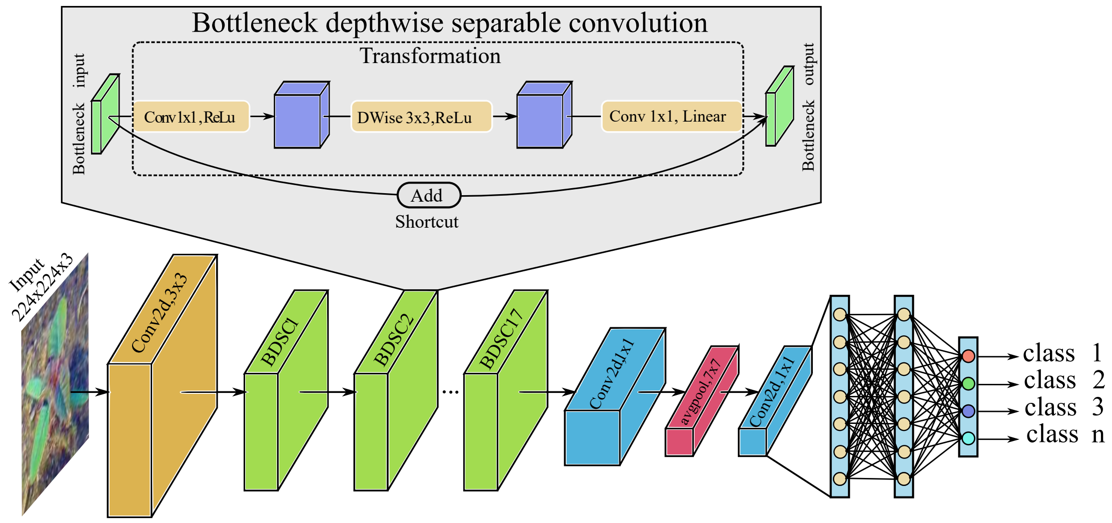


Figure 3.22: Representation of MobileNetV2 architecture. The green blocks mean the Bottleneck Depthwise Separable Convolution blocks (BDSC).

The BDSC blocks are who characterize this network from the other ones (Figure 3.22). A BDSC takes as input a bottleneck input tensor with k channels. First, a *pointwise convolution* ($1 \times 1\, Conv$), followed by a *ReLu6* activation function is used to expand the bottleneck feature map to a higher dimensional space. Then, to re-increase this feature map to a higher dimensional tensor, a *depthwise convolution* is performed using filters of size 3×3 and followed by a *ReLu6* activation function. Next, the output feature map is reduced to a low dimensional space by applying a *pointwise convolution*. Nonetheless, to avoid the loss of information, this last *pointwise convolution* is followed by a *linear activation function*. Finally, the input bottleneck feature map and the final feature map are concatenated to form an integrated output bottleneck feature map. This concatenation strategy is important to avoid the vanishing gradient problem.

The advantage of implementing depthwise separable convolution (*depthwise convolutions* and *pointwise convolutions*) is the reduction of trainable parameters, which comes along with the reduction of computation costs. Sandler et al. (2018) declared that the computation costs of MobileNetV2 network could be 8 to 9 times smaller than that of networks of standard convolutions.

3.1.4.2. Experimental setup using segmentation and classification networks for detection

As previously stated, the suggested detection method comprised two phases: the segmentation stage, relying on a U-Net-like-ResNet101, and the classification stage, involving the implementation and evaluation of ResNet101, VGG16, Xception, and MobileNetV2 networks.

All architectures were trained using a desktop computer equipped with a Core i7 processor, 32 GB of RAM, and an NVIDIA GPU GeForce RTX 3070Ti (8GB). The implementation was conducted in Python 3.8 using the Keras framework with Tensorflow 2.5.0 as the backend.

U-Net-like training

Five set of experiments were conducted looking to derive a strong U-Net-like-ResNet101 model capable of adapting to unseen images. Four experiments were conducted using resized images, and one experiment involved dividing images solely into patches, as detailed in Table 3.2. For this set of experiments, we utilized the large dataset for the visible spectrum containing meticulously annotated images, along with RGB images derived from the multispectral dataset (Figure 3.10). Additionally, the NIR channel of the multispectral dataset was also used (Figure 3.9). However, instances in the Crop and NLW classes in both the RGB dataset and the NIR dataset were augmented to match the number of instances in the BLW class. The splitting rates of both the of the visible spectrum dataset and NIR dataset is also provided in Table 3.2.

Notice in Table 3.2 that the experiments for training the U-Net-like architecture, additional to the image configuration, involved the type of dataset and its configuration. In all the cases, the trained models were tested in the visible spectrum dataset. A transfer learning strategy was applied for all experiments, importing the weights of the convolutional layers from ResNet101 (encoder) when it was originally trained on the ImageNet dataset (Deng et al., 2009), and then these weights were frozen. However, for experiment two, the ImageNet weights of the encoder were used for initializing the training and syntonized with the NIR dataset. Subsequently, in experiment three, the weights were used for the encoder (ResNet101) on the U-Net-like architecture.

Table 3.2: Set of experiments carried out for training the U-Net-like architecture.

		NIR channels		**Visible spectrum dataset**			
Image configuration	**Experiment**	**Training (90 %)**	**Validation (10 %)**	**Training (80 %)**	**Validation (10 %)**	**Test (10 %)**	**Tranfer learning**
Resized image	1			✓	✓	✓	ImageNet
	2	✓	✓			✓	ImageNet*
	3			✓	✓	✓	NIR (Exp. 2)
	4	✓	✓	✓	✓	✓	ImageNet
Patched image	5			✓	✓	✓	ImageNet

*The encoder weights of the U-Net-like architecture were syntonized.

For training the U-Net-like in the first four experiments, the input images were resized to 512×512. On the other hand, for experiment five, when images were divided in patches, the original size of the images was used. However, in this case, image padding was implemented as a pre-processing step during the training and then divided in patches of 512×512 pixels. This ensured that the original size of input images remained unaltered, with pixels of value 0 added on two sides to achieve fixed-size patches. The chosen loss function throughout was the dice loss, known for its strictness in segmentation tasks as it penalizes predominant pixels in specific classes. The calculation of the dice loss was specified in Section 2.3.1.3.

The fine-tuning of the hyperparameters consisted in adjusting the learning rate, optimizer, and the number of epochs. In this way, it was noted that the Adam optimizer with a learning rate of 0.0001 demonstrated better compatibility with our dataset. The model was trained for a total of 100 epochs in the two methods.

The metrics DSC, IoU, and mIoU were used to evaluate the performance of U-Net-like-ResNet101 architecture in this approach.

Training of the classification CNNs

We used the balanced dataset derived from the dataset which encloses all the visible spectrum images (RGB images), which was given in Figure 3.10. As the detection strategy encompassed training a classification CNNs, a subset of data was generated. This subset comprised isolated plant images extracted from the original experimental dataset (images with multiple plants). For training both the U-Net-like and classification networks, the dataset was split into 80 % training, 10 % validation and 10 % test.

The convolutional layers of the classification networks retained their original architectures, while the FCL were customized. Similar to the segmentation network, the initial step involved configuring parameters and hyperparameters of these architectures to suit our dataset. Initially, the weights of both the convolutional and FCL were initialized randomly and trained. Subsequently, the convolutional layers' weights were initialized with those from pre-training on the ImageNet dataset and then retrained with our dataset. Finally, the convolutional layers' weights were initialized with those from ImageNet and then frozen, indicating that only the FCL underwent training.

Respecting the FCL, they were modified from two layers to three layers, with the number of neurons ranging from 512 to 4,096, increasing by 512 for the first and second layers. The ReLU activation function was consistently configured for these initial two layers. The third layer (output) always consisted of three neurons with a softmax activation function, aligning with the classes in our dataset (Crop, NLW, and BLW). Throughout the fine-tuning process, the optimizer, learning rate, loss function, and the number of epochs underwent variations. The dimensions of the input images for all networks were consistently 224×224 pixels.

Evaluation of performance

The evaluation of semantic segmentation, which involves classifying pixels into the classes Crop, NLW, and BLW for plant species, utilized metrics DSC, IoU, mIoU. Whereas for evaluating the performance of the classification networks it were used the metrics accuracy, precision, recall and F1 score. The mathematical expressions of all these metrics were provided in Section 2.4.

3.1.5. End-to-end segmentation based on transformers

In recent studies, the performance of transformer architectures has surpassed that of CNNs for the classification, detection and segmentation of weeds and crop plants. Reedha et al. (2022) reported a superior performance of the visual transformers (ViT) (Vaswani et al., 2017) over that of the architectures EfficientNet and ResNet in classifying herbs over beet, parsley, and spinach crops. Also, Wang et al. (2023) found a superior performance of Swin Transformer (Liu et al., 2021) concerning the well-known CNN VGG-16, ResNet-50, DenseNet-121, SE-ResNet-50, and EfficientNetV2 on the classification of corn seedlings and seven weed species. On the other hand, concerning detection, Zhou et al. (2022) found that a Multi-Window Swin Transformer obtained a better mean Average Precision (mAP) than that of the frameworks Faster R-CNN, Mask R-CNN, FCOS, ATSS, SSD, CenterNet, and YOLOv3 for detecting wheat spikes. Finally, with regard to segmentation, Jiang et al. (2023) evaluated the architectures Swin Transformer, SegFormer, and Segmenter for segmenting ten weed species commonly found in turfgrass fields. In a separate study, Xu et al. (2023) focused on segmenting various weed species in natural soybean fields. They reported higher mIoU for these transformer-based architectures compared to common segmentation models that rely solely on convolutions.

Therefore, in this thesis, realizing that practically no work has been carried out on the segmentation of corn plants and weeds under natural conditions. We explored the transformer architectures Swin-UNet (Cao et al., 2021), Segmenter (Strudel et al., 2021) and SegFormer (Xie et al., 2021) of classifying the pixels belonging to the plant species of the classes Crop, NLW and BLW.

3.1.5.1. Transformers architectures

This Section presents a brief description and functionality of the transformer architectures Swin-UNet, Segmenter and SegFormer.

Swin-UNet

The Swin-UNet model is a fusion of two renowned architectures, namely the Swin Transformer (Liu et al., 2021) and the U-Net (Ronneberger et al., 2015).

The U-Net model is a CNN that specializes in segmentation. The architecture and functionality seem to that provided previously for U-Net-like in section 3.1.3.1. Whereas the Swin Transformer is an adapted version of the vision Transformer (ViT) (Dosovitskiy et al., 2021), which is the first proposed transformer for image classification.

The Swin-UNet, whose architecture is represented in Figure 3.23, consists of an encoder, bottleneck, an encoder and skip connections. As observed, this transformer is consolidated for the Swin Transformer blocks. Each of these blocks contains two consecutive sub-blocks constructed with Layer Norm (LN), either window multi-head self-attention module (W-MSA) module or shifted window-based multi-head self-attention (SW-MSA) module, residual connections, Layer Norm (LN), and an MLP with GELU non-linearity, defined as:

$$\hat{z}^l = W - MSA(LN(z^{l-1})) + z^{l-1} \tag{3.10}$$

$$z^l = MLP(LN(\hat{z}^l)) + \hat{z}^l \tag{3.11}$$

$$\hat{z}^{l+1} = SW - MSA(LN(z^l)) + z^l \tag{3.12}$$

$$z^{l+1} = MLP(LN(\hat{z}^{l+1})) + \hat{z}^{l+1} \tag{3.13}$$

where $\hat{z}^l$ is the output features of either $W - MSA$ or $SW - MSA$ of module one and z^l is the output features of MLP of module one, of block l; $W - MSA$ refers to a regular window multi-head self-attention and $SW - MSA$ is a multi-head self-attention computed with shifted windows. The self attention is computed using the Equation 2.52.

Segmenter

The Segmenter transformer is also an encoder-decoder architecture for semantic segmentation, as shown in Figure 3.24. The encoder is inspired in the ViT transformer (Dosovitskiy et al., 2021) while the encoder is based-mask inspired in the DEtection TRansformer (DETR) (Carion et al., 2020).

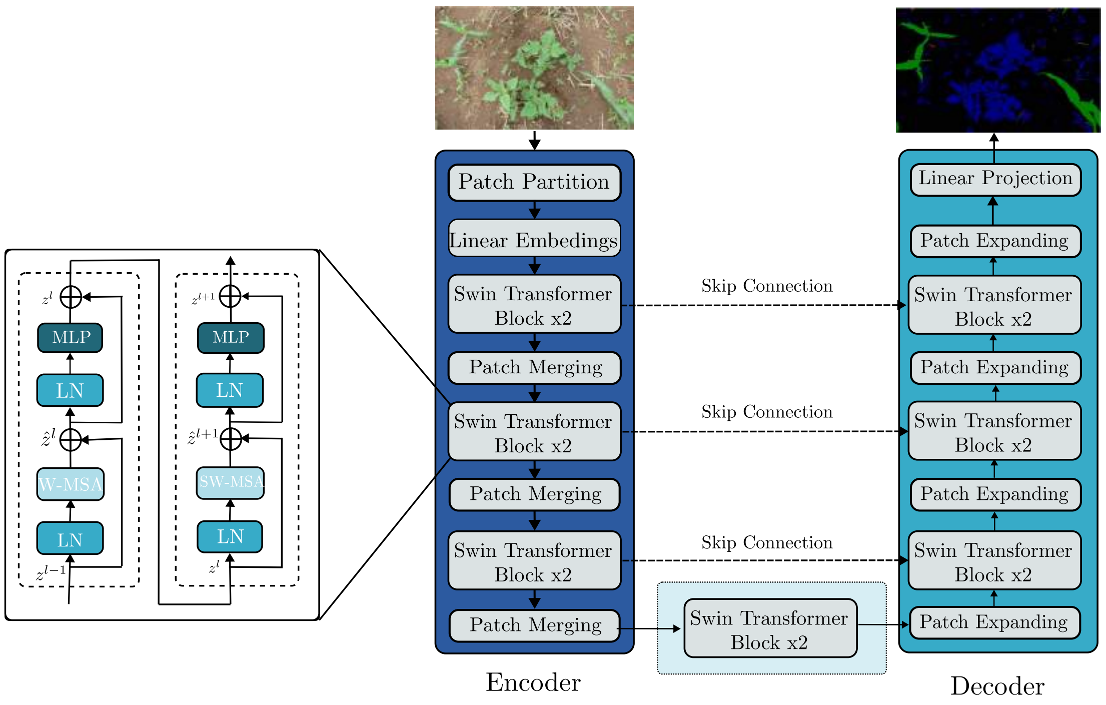


Figure 3.23: Illustration of the Swin-UNet transformer architecture.

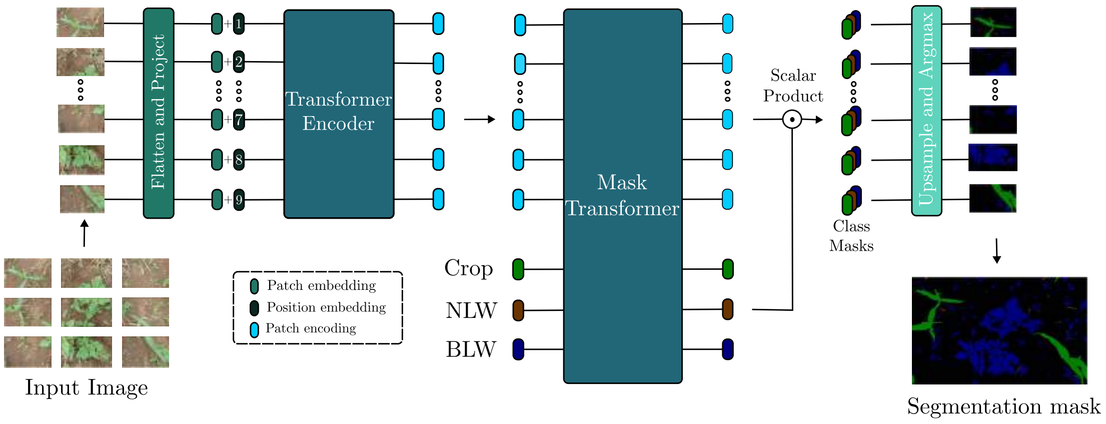


Figure 3.24: Representation of the segmenter transformer architecture.

The input image for this transformer $I \in \mathbb{R}^{H \times W \times C}$ undergoes division into N fixed-size patches, forming a sequence of patches $x = [x_1, ...x_N] \in \mathbb{R}^{N \times p^2 \times C}$. Here $N = \frac{HW}{p^2}$, $p^2 := (p, p)$ refers to the patch size, and C is the number of channels. Subsequently, each patch in the sequence x is flattened into a 1D vector and linearly projected to a patch embedding to produce the sequence $x_0 = [Ex_1, ..., Ex_N] \in \mathbb{R}^{N \times D}$ where $E \in \mathbb{R}^{D \times (p^2 C)}$. Then position embedding $pos = [pos_1, ..., pos_N] \in \mathbb{R}^{N \times D}$ is added to the patch embedding to get the input sequence of tokens $z_0 = x_0 + pos$.

Each layer L of the transformer is applied token sequence z_0 to produce a sequence of contextualized encodings $z_L \in \mathbb{R}^{N \times D}$. Each transformer layer is composed of a multi-head self-attention (MSA) block followed by a layer norm (LN), then a point-wise MLP followed by a layer norm (LN) and residual connections, expressed as,

$$a_{i-1} = MSA(LN(z_{i-1}) + z_{i-1} \tag{3.14}$$

$$z_i = MLP(LN(a_{i-1}) + a_{i-1} \tag{3.15}$$

where $i \in \{1, ..., L\}$. The attention mechanism is computed also with Equation 2.52. Therefore, the purpose of the segmenter is to decode every patch embedding z_L into a segmentation map $s \in \mathbb{R}^{H \times W \times K}$, where K is the number of classes.

SegFormer

The SegFormer transformer (Xie et al., 2021), designed for semantic segmentation tasks, comprises an encoder and a decoder section, as illustrated in Figure 3.25. The encoder of this network exhibits the capability to handle various image sizes without compromising the overall performance of the architecture. This block incorporates hierarchical transformers, that enable the creation of both high-resolution fine feature maps and low-resolution coarse feature maps.

Conversely, the SegFormer encoder includes a lightweight MLP responsible for processing high-resolution feature maps that capture local properties extracted by the lower layers of the encoder. Simultaneously, the MLP processes low-resolution coarse features containing global features extracted by the deeper layers of the encoder. Consequently, the MLP decoder combines both global and local attention characteristics from input images, resulting in dominant segmented objects.

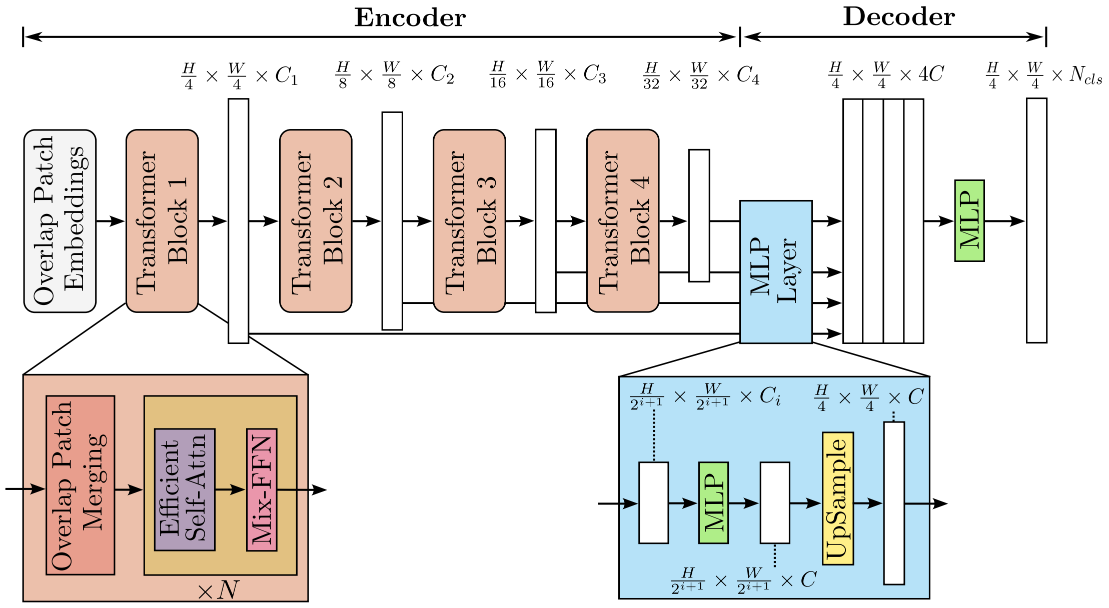

Encoder
Decoder
$\frac{H}{4} \times \frac{W}{4} \times C_1$
$\frac{H}{8} \times \frac{W}{8} \times C_2$
$\frac{H}{16} \times \frac{W}{16} \times C_3$
$\frac{H}{32} \times \frac{W}{32} \times C_4$
$\frac{H}{4} \times \frac{W}{4} \times 4C$
$\frac{H}{4} \times \frac{W}{4} \times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Overlap Patch Merging
Efficient Self-Attn
Mix-FFN
$\times N$
$\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}} \times C_i$
$\frac{H}{4} \times \frac{W}{4} \times C$
MLP
UpSample
$\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}} \times C$

Figure 3.25: Representation of the SegFormer transformer architecture.

Specifically, when the encoder is provided, for example, with an image of dimensions $H \times W \times 3$, it initially divides it into patches of size 4×4. Subsequently, the hierarchical transformers generate hierarchical feature maps (F_i) with resolutions $\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}} \times C_i$, where $i \in \{1, 2, 3, 4\}$. Put differently, the hierarchical transformers of SegFormer produce feature maps with sizes $\frac{1}{4}, \frac{1}{8}, \frac{1}{16}$ and $\frac{1}{32}$, respecting the dimensions of the input image.

Each hierarchical transformer within the encoder consists of the sub-modules: Overlapping Patch Merging, Efficient Self-Attention, and Mix-Feedforward Network (Mix-FFN). The role of Overlapping Patch Merging is to reduce the hierarchical feature maps along the encoder. For example, it transforms $F_1(\frac{H}{4} \times \frac{W}{4}) \times C_1$ to $F_2(\frac{H}{8} \times \frac{W}{8}) \times C_2$. The Efficient Self-Attention mechanism conducts attention operations using the heads ($Q, K,$ and V) through the utilization of Equation 2.52.

However, SegFormer introduces a sequence reduction process. In the original multi-head self-attention, the dimensions of the heads ($Q, K,$ and V) are $N \times C$ (Vaswani et al., 2017), where $N = H \times W$ represents the length of the sequence. This uniformity in dimensions complicates the analysis, especially for large-sized images, due to a computational complexity of $O(N^2)$. To address this challenge, SegFormer employs a ratio R to reduce the length of the sequence K as follows,

$$\hat{K} = Reshape\left(\frac{N}{R}, C \cdot R\right)(K) \tag{3.16}$$

$$K = Linear(C \cdot R, C)(\hat{K}) \tag{3.17}$$

In this way, the reduced K has dimensions $\frac{N}{R} \times C$. Due to this strategy, the complexity of the self-attention mechanism is reduced to $O\left(\frac{N^2}{R}\right)$, instead of the original complexity of $O(N^2)$.

Finally, within each hierarchical transformer of the encoder, the Mix-FFN submodule addresses data-driven positional encoding. To achieve this, a $3 \times 3\, Conv$ integrates the Feed-Forward Network (FFN), imparting an effect of no padding to escape location information. The calculation for Mix-FFN is as follows:

$$x_{out} = MLP(GELU(Conv_{3\times3}(MLP(x_{in})))) + x_{in} \tag{3.18}$$

where, x_{out} is the output feature from the Mix-FFN module, x_{in} is the feature from the self-attention module and GELU is the Gaussian Error Linear Units activation function.

The decoder section exclusively consists of MLP layers, operating on features derived from the hierarchical transformer blocks of the encoder module. To achieve the final segmentation, this decoder performs four primary steps. (1) The multi-level features F_i from each transformer block of the encoder pass through an MLP layer, unifying them in their channel dimension. (2) The resulting feature maps are then upsampled to $\frac{1}{4}$ of their resolution and concatenated together. (3) An additional MLP layer concatenates the upsampled features F. (4) Finally, a separate MLP processes this fused feature map to predict the ultimate segmentation mask M, with dimensions$\frac{H}{4} \times \frac{W}{4} \times N_{cls}$, where N_{cls} denotes the number of classes in the training dataset.

3.1.5.2. Experimental setup for segmenting crop and weeds based on transformers

For training the transformers Swin-UNet, Segmenter and SegFormer, the large dataset shown in Figure 3.10, derived from the dataset that encloses all the visible spectrum images (RGB images), was used. This dataset grouped the classes Crop, NLW and BLW. However, it was first augmented the instances of the classes Crop and NLW to the number of instances of the class BLW to obtain a balanced dataset. Then, it was split into 80 % for training, 10 % for validation and 10 % for testing the trained models.

To train the models, a desktop computer with Core i7 processor, NVIDIA GPU GeForce GTX 1080Ti with 6 GB of VRAM, and 32 GB of RAM memory has been used. The implementation was carried out in Python 3.8 and PyTorch framework 1.13.1.

The images of the dataset for training the three transformers were resized to a dimension of 512×512 and then their normalized pixels into the range $[0, 1]$. The Swin-UNet architecture was trained using the Adam optimizer with a learning rate of 0.00001, a batch size of one and categorical cross-entropy loss function. For the Segmenter, the stochastic gradient descent (SGD) optimizer with 0.001 learning rate was used, also a batch size of one and categorical cross-entropy loss function were used. Finally, the SegFormer architecture was trained using the Adam optimizer with a learning rate of 0.00001, a batch size of one, and categorical cross-entropy loss. The three models were trained for 100 epochs.

Evaluation of performance

The evaluation of the trained transformer models in classifying the pixels of the classes Crop, NLW, and BLW of the dataset was carried out utilizing the metrics DSC, IoU and mIoU.

3.2. Mechatronic platform development

In this thesis, DL models have been trained to detect crop plants, narrow-leaf weeds, and broad-leaf weeds in authentic corn fields. As mentioned earlier in this Chapter 3, the vision system would be evaluated for its ability to control weeds by spraying herbicides. Therefore, we have the following problem to solve,

Problem 1. Given a vision system that detects corn plants (Crop), NLW, and BLW, the problem is to develop a platform commanded for the vision system that could navigate in an authentic cornfield spraying herbicides to control NLW and BLW in real-time.

At first, the platform should be experimental and low-cost, utilizing materials and electronic components available in the Autonomous and Intelligent Systems Laboratory of the Research Center in Optics. One notable constraint involves navigation within natural cornfields; the device should be capable of adapting to crop rows spaced between $0.70m$ to $0.80m$ apart from each other. Under this premise, the conceptual design of the platform involves exclusively developing a morphological chart that presents various alternatives for the systems, as illustrated in Table 3.3.

Table 3.3: Morphological chart with alternatives for the mechatronic platform.

	Alternatives			
Systems and sub-systems	**1**	**2**	**3**	**4**
Navegation				
Type	Autonomous	Pulled by tractor	–	–
Mechanical design				
Material	Wood	Aluminium	Steel	Plastic
Assembly	Welded	Screwed	–	–
Chemical tank	Plastic	–	–	–
Electrical and hydraulic design				
Power supply	Gasoline generator	LiPo battery	Supplied from tractor	Car battery
Controller	Arduino	PIC	Raspberry Pi	–
Solenoid valve	Hydraulic activation	Pneumatic activation	Electric activation	
Nozless	Flat fan	Solid cone	Hallow cone	Adjustable
Pump	Diaphragm (Mechanical)	Diaphragm (Electrical)	–	–
Vision system release				
Processing type	*insitu*	Remote		
Camera type	Smartphone	Professional	Webcam	

After analyzing the alternatives and restrictions, the components of the systems were selected. Then, the platform was drawn in the software SolidWorks 2019, constructed, and instrumented. The weed detection model was carefully linked to the actuators and also a Graphical User Interface (GUI) was designed to facilitate the operation of the smart sprayer in the field. The redesign step was also necessary after some tests in real field conditions.

3.3. Field evaluation

The field evaluation of the intelligent sprayer was conducted at the Pabellon Experimental Extension of the National Institute for Research in Forestry, Agriculture and Livestock (INIFAP, by its acronyms in Spanish) in Aguascalientes, Mexico, located at 22° 11' N and 102° 20' W, at an elevation of 1912 meter over sea level.

A corn crop field of size 50×60 meters was established on September 29, 2023. The soil was conventionally prepared on September 20, with a pass of a disc plow and two cross passes of harrow. Due to the relatively dry soil conditions, furrowing the field was necessary for subsequent gravity irrigation to ensure the germination of corn seeds. On the day of sowing, the soil was disked again to eliminate furrows and smooth the surface. The seeder was calibrated to obtain a corn plant density of around $100,000\,plant\,ha^{-1}$ with distance among plant rows of $0.77m$. Subsequently, immediately after the sowing practice the pre-emergence herbicide $Acetochlor + Atrazine$ was sprayed at a rate of $310g + 123g\ i.a\,ha^{-1}$, because the field area had a variety of mature weeds species disposed of in high density, which had plenty of viable seeds. The objective was to prevent a high density of weeds during the experimental stage and thereby observe the discrimination effect of our smart sprayer. Subsequent management practices involved solely gravity irrigation to encourage the growth of both the crop and weeds.

The evaluated treatments were our SWS and a Swissmex conventional sprayer ($Model\,890006$) (CWS), which served as the control treatment. The CWS is equipped with 21 flat fan nozzles, a tank with a capacity of $600L$, and an aspersion width of $10.5m$. The response variables measured were the herbicide expenditure per hectare ($L\,ha^{-1}$) and the effectiveness in controlling NLW and BLW in the experimental corn crop. The treatments were distributed in a randomized complete block design with three replications. The experimental unit for the SWS comprised three consecutive $50m$ long passes, while for CWS, it involved a single $50m$ long pass. A broad-spectrum post-emergence herbicide, $Mesotrione + Atrazine$, was sprayed once in rates for treatment as shown in Table 3.4. The experiment took place on November 19, 2023.

Table 3.4: Experimental treatments and pos-emergence herbicide rates.

Treatment	**Herbicide**	**Rate** ($g\,i.a\,ha^{-1}$)	**Spray volume** ($L\,ha^{-1}$)
Conventional sprayer (CWS)	*Mesotrine + Atrazine*	$180 + 1,151$	600
Smart weed sprayer (SWS)	*Mesotrine + Atrazine*	$180 + 1,151^{*}$	RV
		$120 + 767^{+}$	RV

*Herbicide dose to control NLW weeds. +Herbicide dose to control BLW weeds. RV– Response variable.

As observed in the previous Table 3.4, the SWS sprayed different doses of the same herbicide. This variation is due to its dual-tank system, with one tank storing chemicals for controlling NLW and the other for BLW. It's worth noting that the herbicide rate for the SWS treatment maintained the same diluted concentration as the CWS treatment.

It is worth mentioning that, prior to the experimental practice, the weed density per square meter ($plt\,m^{-2}$) was quantified by tanking six samples along the main diagonals of the cornfield. Additionally, the plant species in the experimental cornfield also were registered.

To quantify the response variable of herbicide expenditure per hectare ($L\,ha^{-1}$), both the CWS and SWS tanks were carefully filled using $5L$ and $1L$ test tubes. The same procedure was repeated after the spraying practice to measure the remaining herbicide mixture.

Effectiveness in controlling NLW and BLW was assessed using image analysis by estimating the reduction of weed cover respecting the time. For this variable, the observation unit consisted of the two central rows of each treatment, which were marked for subsequent analysis. Immediately after spraying, images with a resolution of $4,608 \times 3,456$ were captured using a drone (DJI Parrot ANAFI) flying at a height of $5m$ above the soil surface. Then, two subsequent captures were conducted every five days to observe the chemical actions.

The individual images captured in each observation unit were combined to generate an orthomosaic. This process was executed in Python 3.8, utilizing the widely recognized scale-invariant vision transform (SIFT) descriptor (Lowe, 1999) and the random sample consensus algorithm (RANSAC) (Fischler and Bolles, 1981) to ensure accurate image overlap. Nonetheless, the software Photoshop was also employed for this task, particularly for creating orthomosaics from images with a limited overlap area. Subsequently, all the orthomosaics were adjusted in size according to the crop row width using Python 3.8.

After configuring the orthomosaics, they were divided into smaller images and the weeds were manually annotated through the tool VGG Image Annotator (Dutta and Zisserman, 2019). This practice follows the broad annotation procedure described in Section 3.1.1.2 to obtain weed masks for each observation unit. Finally, the number of pixels in each observation unit was transformed into a percentage, with the observations obtained on the day of the experiment considered as 100 %.

3.3.1. Statistical analysis

The observed field observations, for both herbicide expenditure and effectiveness in controlling weeds, were subjected to an analysis of variance under a randomized complete block design and Tukey's multiple comparison test ($P \leq 0.05$). For all the analysis, the R software (3.6.2) was used.

Chapter 4

Results and discussion

As declared earlier, this thesis aimed to develop a real-time vision system for detecting and controlling weeds in authentic corn fields.. To better order and understand the finding, this Chapter is divided in results and discussion, provided in Section 4.1 and Section 4.2, respectively. Section 4.1.1 of the results gives the findings obtained from each of the explored strategies for the vision system development. Subsequently, Section 4.1.2 offers the outcomes of the designed and instrumented intelligent platform for controlling weeds under authentic corn fields. Finally, the experimental field evaluation of the smart sprayer is presented in Section 4.1.3.

4.1. Results

4.1.1. Vision systems development

This section presents the strategies explored for developing the vision system. Section 4.1.1.1 recapitulates the created dataset. Section 4.1.1.2 provides the results of the classification of plant species based on shallow and deep learning. Section 4.1.1.3 shows the segmentation of our dataset based on R-CNN and U-Net-like architectures. Subsequently, Section 4.1.1.4 offers the findings observed in the strategy of combining segmentation and classification networks to achieve detection. Finally, Section 4.1.1.5 shows the segmentation of species based on pure transformers.

4.1.1.1. Annotated dataset

To train a DL architecture, a vast quantity of images captured at different scenarios and growing stages of the plants in real field conditions are needed so that they can generalize to unseen data. The images of our dataset were captured during the spring-summer agricultural cycle, as it was declared. In total $15,885$ RGB images were acquired using different camera devices. Meanwhile $2,312$ images were acquired using a multispectral camera with five bands (R, G, B, NIR, and RedEdge). Images were captured in different cornfield locations and different growth stages of the plants. The sunlight variability and the natural background in each image were also introduced.

A total of $10,575$ RGB images out of the $15,885$ were annotated at the pixel level, following a careful annotation using polygons. The annotated plant species were Crop plant (*Zea mays*); and the weeds, *Cynodon dactylon*, *Eleusine indica*, *Digitaria sanguinalis*, *Cyperus esculentus*, *Portulaca oleracea*, *Tithonia tubaeformis*, *Amarantus spinosus* and *Malva parviflora*. The total number of annotated instances per plant species was provided in Figure 3.7. Subsequently, the $2,312$ multispectral images were also annotated at the pixel level, initially with a coarse annotation using polygons, followed by a post-treatment using digital image analysis. The annotation of multispectral images included corn plants (Crop) and multiple NLW or multiple BLW. The number of annotated instances in the group classes Crop, NLW and BLW is shown in Figure 3.9.

Therefore, in total, by combining the RGB images acquired with visible cameras and the multispectral camera, a dataset of $12,887$ annotated images is obtained. Grouping the plant species from the two types of images into the classes Crop, NLW and BLW, a total number of $28,507$, $41,795$ and $52,541$ instances, respectively, are obtained per class. Therefore, our dataset could give new models the potential to be transferred to natural corn field applications once they are trained on it. This dataset is depicted in Figure 3.10.

4.1.1.2. Classification based on shallow and DL

The results of classifying the plant species of our dataset in the classes Crop, NLW and BLW by using the rotation-invariant uniform local binary pattern +

support vector machine ($LBP_{P,R}^{riu2}, SVM$) and classification CNN are presented in this Subsection.

Classical machine learning approach

A series of experiments were conducted to assess the effectiveness of the proposed classical ML approach (SVM), as depicted in Table 3.1. As illustrated, the LBP operator was employed with three distinct spatial and angular resolutions, denoted as (P, R), featuring values of $(8, 1)$, $(16, 2)$, and $(24, 3)$. Moreover, three varying image dimensions (256×256, 128×128, and 64×64 pixels) underwent testing. Depending on the size, these images were subsequently subdivided into cells with dimensions of 8×8, 16×16, 32×32, 64×64, and 128×128.

The performance of the classifiers is detailed in Table 4.1. Notably, the top three SVM models were configured as follows: $LBP_{8,1}^{riu2}/\ 256 \times 256/32 \times 32$, $LBP_{24,3}^{riu2}/256 \times 256/32 \times 32$, and $LBP_{24,3}^{riu2}/128 \times 128/32 \times 32$. These configurations achieved corresponding accuracy rates of 83.04 %, 82.76 %, and 82.26 % over the test data, respectively. These percentage values indicate the proportion of plant species correctly classified into their respective classes. Additionally, these models exhibit consistent performance across the metrics of precision, recall, and F_1 score, with differences among these variables for each of the three models being less than one order of magnitude, as shown in Table 4.1. Concerning the test time of these three models, the configuration $LBP_{24,3}^{riu2}/128 \times 128/32 \times 32$ demonstrated a slightly shorter duration, with a $1.89ms$ difference compared to the model with the highest accuracy.

Additionally, Figure 4.1 shows the mean accuracy for each of the $LBP_{P,R}^{riu2}$ texture features. The accuracy value is practically consistent among the exact image size for the three $LBP_{P,R}^{riu2}$ operators, with differences of less than a unit of magnitude for this variable. Furthermore, the mean accuracy for the image size 256×256 was slightly superior to the other two sizes for each $LBP_{P,R}^{riu2}$ operator. The metrics precision, recall, and F_1 score also showed the same behavior for the image size 256×256 in each computed image operator.

Table 4.1: Performance of SVM models in classifiying trained with $LBP_{P,R}^{riu2}$ texture features.

		256 × 256 Image					**128 × 128 Image**				**64 × 64 Image**		
		Cell Size					**Cell Size**				**Cell Size**		
(P, R)	**Metrics**	**8 × 8**	**16 × 16**	**32 × 32**	**64 × 64**	**128 × 128**	**8 × 8**	**16 × 16**	**32 × 32**	**64 × 64**	**8 × 8**	**16 × 16**	**32 × 32**
	Accuracy (%)	79.39	79.39	**83.04***	81.50	79.28	77.34	80.31	81.82	80.35	77.60	78.32	77.01
	Precision (%)	79.82	79.73	82.94	81.44	78.95	77.50	80.23	81.65	80.22	77.64	78.22	76.74
(8,1)	Recall (%)	79.40	79.41	82.90	81.59	79.16	77.30	80.22	81.72	80.31	77.75	78.27	76.93
	F_1 score (%)	79.54	79.54	82.91	81.50	79.03	77.39	80.21	81.65	80.21	77.68	78.24	76.81
	Test time (ms)	407.65	235.57	212.07	206.46	205.03	227.98	211.58	204.57	200.28	205.70	199.77	198.25
	Accuracy (%)	79.72	79.3	81.36	79.96	76.5	77.54	80.53	79.65	78.45	77.08	80.31	76.73
	Precision (%)	80.02	79.96	81.23	79.69	76.03	77.69	80.70	79.72	78.52	77.07	80.27	76.71
(16,2)	Recall (%)	79.75	79.40	81.34	79.82	76.37	77.29	80.64	79.88	78.63	77.14	80.33	76.05
	F_1 score (%)	79.87	79.57	81.27	79.74	76.13	77.42	80.67	79.73	78.58	77.10	80.30	76.86
	Test time (ms)	439.53	250.37	221.68	215.32	213.08	241.71	210.24	202.42	202.20	207.66	200.71	199.02
	Accuracy (%)	77.07	80.77	**82.76**	81.27	78.12	77.93	81.34	**82.26**	77.62	76.99	78.78	77.16
	Precision (%)	77.56	81.20	82.80	81.10	77.69	78.17	81.40	82.13	77.60	77.30	78.73	77.79
(24,3)	Recall (%)	77.08	80.82	82.77	81.20	77.90	77.90	81.38	82.18	77.76	77.02	78.93	77.10
	F_1 score (%)	77.19	80.97	82.78	81.14	77.77	78.01	81.38	82.14	77.67	77.11	78.76	76.90
	Test time (ms)	461.86	263.72	228.92	220.50	217.75	257.94	217.10	210.18	203.83	220.46	205.14	200.28

* Highlighted accuracies correspond to the best three SVM models.

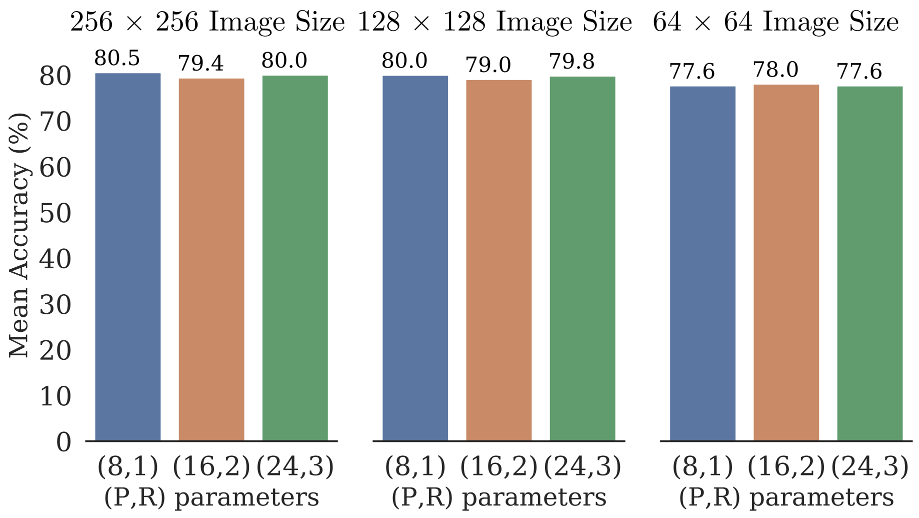


Figure 4.1: The mean of the accuracy for each of the (P, R) defined parameters.

Classification based on CNNs

We utilized the VGG16, VGG19, and Xception architectures for this study. The convolutional layers of these networks were retained, while their FCL were customized to classify the plant species in our dataset. The configuration of the FCL included an input layer and an output layer. The input layers were equipped with 512 neurons and a ReLU activation function. A dropout regularization rate of 0.5 was applied to this layer. Given our three classes, Crop, NLW and BLW, the output layers consisted of three neurons, followed by the *Softmax* activation function.

The performance trends of the accuracy and loss functions for VGG16, VGG19, and Xception during the training stage are illustrated in Figure 4.2. As evident, starting from epoch one, there is a noticeable rise in accuracy and a significant drop in the error value for each of the three models. This pattern is attributed to the implemented transfer learning, a technique known for inducing rapid convergence in models (Espejo-Garcia et al., 2020). In transfer learning, the weights of the convolutional layers, pre-trained on a different dataset, are retained, and only the final layers are adapted to the new data.

VGG16 and VGG19 achieved stability in accuracy for both the training and validation data at epochs 39 and 45, respectively. On the other hand, Xception's accuracy exhibited fluctuations throughout the entire training process; however, the magnitude of these fluctuations decreased, especially from epoch 48 onwards.

Regarding the cost function of each model depicted in Figure 4.2b, VGG16 exhibited the least error in the validation data from epoch 70, overcoming VGG19 and Xception. While the error of VGG19 displayed a smooth trajectory starting from epoch 58, it exhibited an increasing trend until epoch 100, indicating overfitting. Similarly, the error of Xception fluctuated throughout the entire training process, making it less conclusive for determining the optimal number of epochs for our dataset. The fluctuations in Xception's error, varying between maximum and minimum values during training, were also noted by Peteinatos et al. (2020). However, it outperformed the fluctuations observed in VGG16 and RestNet-50 when trained with twelve species of plants.

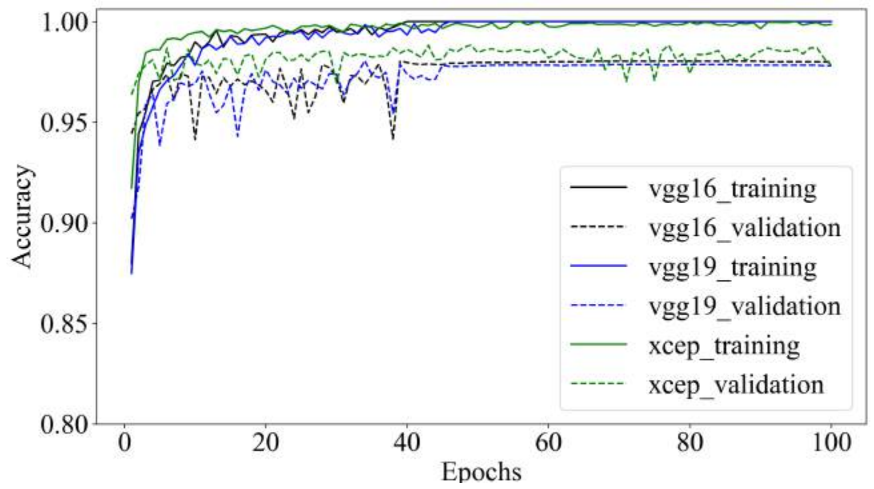
The average performance of VGG16, VGG19, and Xception on the validation data, considering accuracy, precision, recall, F_1 score, and test time, is presented in Table 4.2. The mean values for these metrics ranged from 97 % to 98 %. Overall, VGG16 emerged as the top-performing model with an accuracy of 97.83 %. This ranking was consistent across precision, recall, and F_1 score metrics compared to VGG19 and Xception, with differences consistently less than one order of magnitude in all cases. Xception achieved the best test time, which was 50.18ms faster than VGG16.

Table 4.2: Mean performance of the classification CNN models.

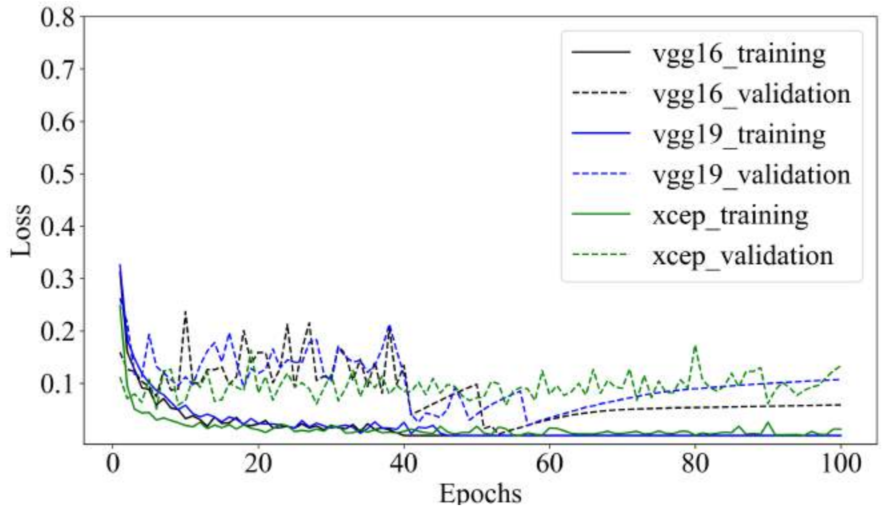
Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F_1 Score (%)**	**Test Time (ms)**
VGG16	97.83	97.67	97.67	97.67	194.56
VGG19	97.44	97.33	97.33	97.33	226.96
Xception	97.24	97.33	97.00	97.00	144.38

Comparison of shallow learning and CNNs

A comparative analysis between the top three classical ML models and the three CNN models is presented in this section. Let SVM$_A$ be the model trained



(a)



(b)

Figure 4.2: Graphs illustrating the training dynamics of VGG16, VGG19, and Xception. (a) Accuracy curves and (b) cost function.

with $LBP_{8,1}^{riu2}/256 \times 256/32 \times 32$, SVM$_B$ the model trained with $LBP_{24,3}^{riu2}/256 \times 256/32 \times 32$, and SVM$_C$ the model trained with $LBP_{24,3}^{riu2}/128 \times 128/32 \times 32$.

Figure 4.3 illustrates the comparison between the three best classic ML approaches and the three CNN models. The average performance of the CNN models surpassed that of the SVM models. For instance, VGG16, the best CNN model, exhibited a mean accuracy that exceeded SVM$_A$, the best classical ML model, by 14.79 %. Furthermore, VGG16 demonstrated a speed advantage, being 1.11x faster than SVM$_A$ in analyzing an image.

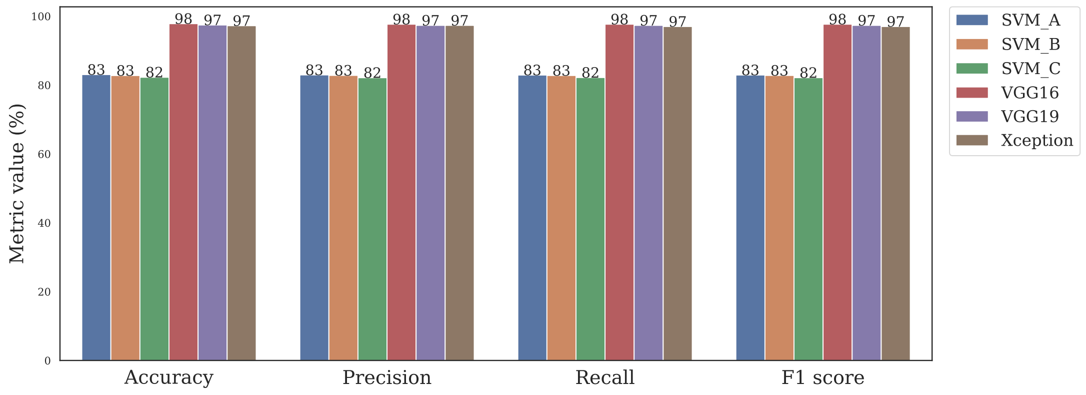


Figure 4.3: Comparison of classification approaches.

Moreover, confusion matrices were generated to assess each model's performance. Figure 4.4 displays the three confusion matrices corresponding to each of the SVM models. The highest accuracy was achieved at 92.4 % for BLW by SVM$_B$ (Figure 4.4b). However, all models encounter challenges when attempting to distinguish between the classes "Crop" and "NLW", often misclassifying instances from one class as the other are exhibited. In the best-case scenario, there is approximately 15 % confusion; in the worst-case scenario, misclassification reaches up to 21 %.

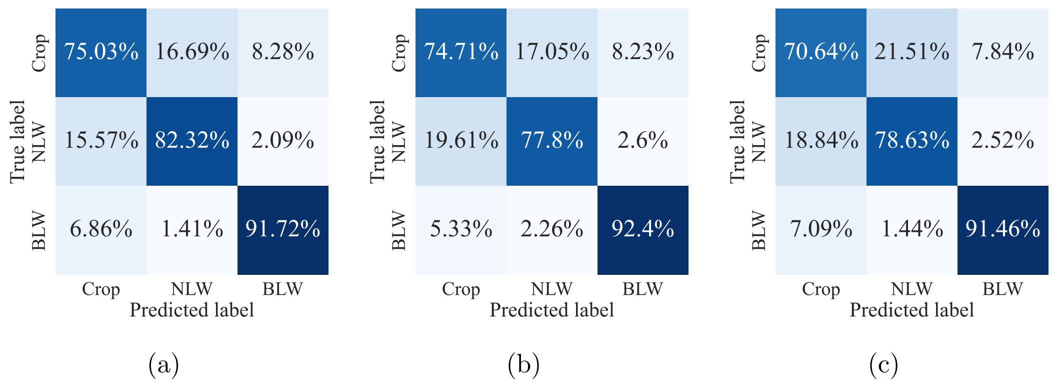


Figure 4.4: Confusion matrices of the three SVM models. (a) SVM$_A$, (b) SVM$_B$, and (c) SVM$_C$.

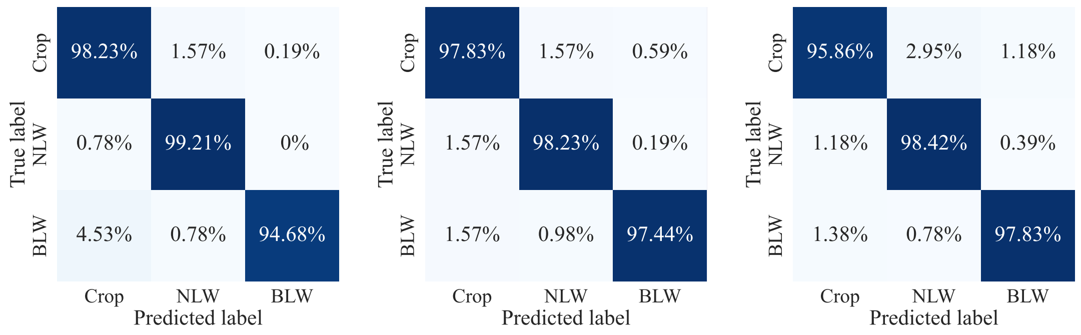
The class with the most accurate identification across all models was BLW, followed by NLW, while Crop consistently exhibited the lowest accuracy. SVM$_B$ achieved the highest identification accuracy for BLW at 92.4 %, as mentioned earlier. In terms of NLW and Crop, both classes were most accurately identified by SVM$_A$, with accuracies of 82.32 % and 75.03 %, respectively. A possible explanation for the confusion between the "Crop" and "NLW" classes is their common membership in the monocot species, sharing numerous texture features.

On the contrary, Figure 4.5 displays the confusion matrices for the CNN models. In this scenario, VGG19 achieved superior classification for Crop and NLW, with 98.23 % and 99.21 % accuracy, respectively. Meanwhile, Xception excelled in classifying BLW with an accuracy of 97.83 %. The VGG16 CNN model, despite having the highest mean accuracy, showcased a more balanced classification across classes, with the maximum difference being only 0.79 % between NLW and BLW. Similar to the SVM models, the CNN models also tended to confuse Crop with NLW and vice versa.

Xception misclassified Crop into NLW at 2.95 %, compared to 1.57 % by both VGG16 and VGG19. VGG16 had the highest misclassification rate for NLW as Crop at 1.57 %, representing the most common misclassification of this class among the models. Additionally, all three models predominantly misclassified BLW more frequently as Crop than NLW. Based on the outcomes, the three CNN models achieved superior performance compared to the results obtained by the three SVM models.

4.1.1.3. Segmentation based on R-CNN and U-Net-like

This section presents the results obtained when our dataset was pixels-wise segmented following two strategies. The first strategy uses RGB images to train the architectures Mask R-CNN-ResNet50, Mask R-CNN-ResNet101, Mask R-CNN-ASPP-ResNet50, Mask R-CNN-ASPP-ResNet101, U-Net-like-ResNet50, and U-Net-like-ResNet101. It is worth noting that the Mask R-CNN-ASPP-based networks were proposed in this thesis.



(a) (b) (c)

Figure 4.5: Confusion matrices of the three CNN models. (a) VGG16, (b) VGG19 and (c) Xception.

Behavior of loss functions and mIoU during training

The loss functions' behavior, of the six architecture configurations through the training and validation data at each epoch is shown in Figure 4.6. The plotted loss curve for the Mask R-CNN and Mask R-CNN-ASPP architectures corresponds to the average binary cross-entropy loss (L_{mask}), which is applied to the segmentation branch of these networks.

The red and blue curves represent the Mask R-CNN-ResNet50 and Mask R-CNN-ResNet101 networks. Conversely, the magenta and cyan curves belong to the Mask R-CNN-ASPP-ResNet50 and Mask R-CNN-ASPP-ResNet101 networks. Lastly, the green and black curves depict the training behavior of the U-Net-like-ResNet50 and U-Net-like-ResNet101, respectively.

Examining Figure 4.6 reveals a consistent trend across all investigated networks, wherein the training and validation errors tend to decrease throughout the training period, signifying effective learning. However, it is notable that the loss curves for Mask R-CNN-based architectures exhibit some fluctuations during the entire training process, while the error curves for U-Net-like models consistently show a monotonically decreasing pattern.

As mentioned earlier, the segmentation result from Mask R-CNN-based models depends on the ROIs detected by the Region Proposal Network (RPN) in the image. At the same time, the U-Net-like model analyzes the entire image.

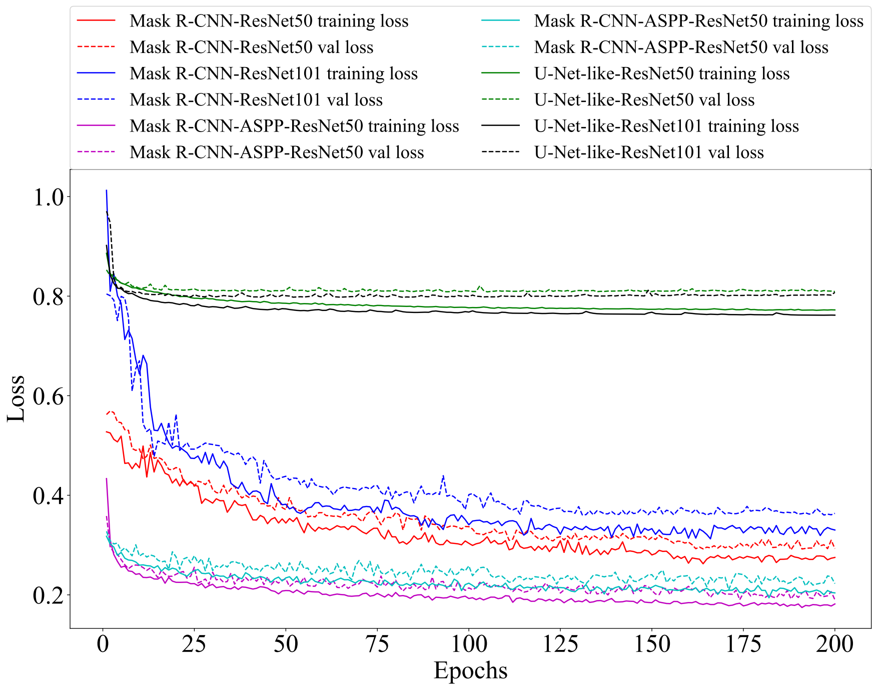


Figure 4.6: The overall behavior of the loss functions for the four networks throughout the training process.

Therefore, the oscillation in the error of Mask R-CNN-based models is attributed to the mask loss function's dependency on the class and box loss functions that the networks operate during training. This concept was also supported by Liu et al. (2020), demonstrating that deep learning detection networks often exhibit high-magnitude detection errors due to the density of plants, as the detected regions contain neighboring leaves or background pixels.

Additionally, as depicted in Figure 4.6, the overall exhibited error of the Mask R-CNN-ASPP architectures is consistently lower during all the training steps compared to the error magnitude of the original Mask R-CNN architectures. Specifically, Mask R-CNN-ASPP-ResNet50 demonstrated the least loss behavior.

The IoU metric provides a direct assessment of the performance of a segmentation model by indicating the overlap between the prediction mask and the ground truth. Therefore, the value of this metric was recorded over the validation data at each epoch. Figure 4.7 illustrates the behavior of the mIoU metric for the six networks. The highest value was achieved by U-Net-like-ResNet101 during the training process, followed by U-Net-like-ResNet50. The mIoU of the Mask R-CNN-ASPP networks consistently surpassed that of the original Mask R-CNN architecture in all epochs. Furthermore, after epoch 75, Mask R-CNN-ASPP-ResNet50 exhibited the highest mIoU, indicating superior performance. In contrast, Mask R-CNN-ResNet101 consistently displayed lower values throughout the training process.

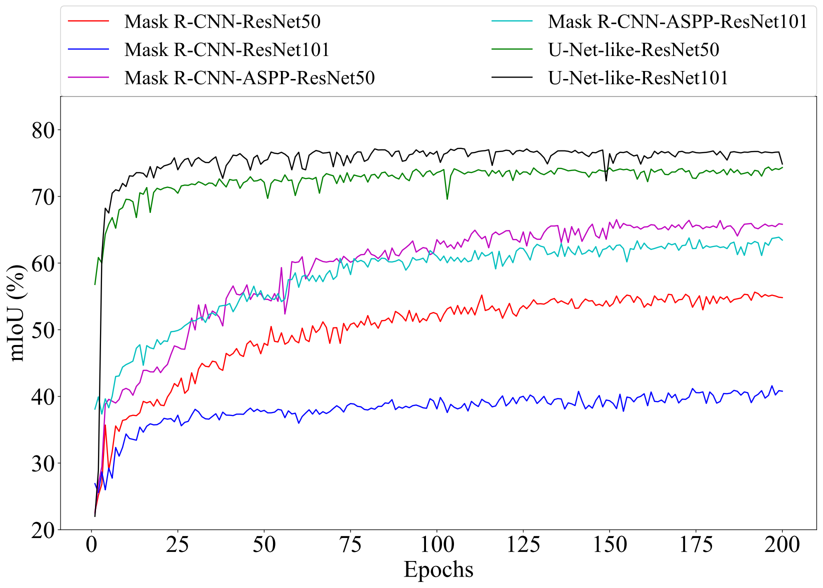


Figure 4.7: Mean intersection over union provided by the models of the four architectures at every epoch during training on the validation data.

Performance of segmentation models over the classes

Table 4.3 presents the performance of the six models in classifying each pixel of the images into the classes Crop, NLW, BLW, and Soil.

The evaluation of performance indicates that the Mask R-CNN-ASPP models surpass those of the original Mask R-CNN models. However, it is noteworthy that the metric values of the Mask R-CNN-ASPP models have been surpassed by those achieved by the U-Net-like models.

Concerning the performance based on DSC metric, the two Mask R-CNN-ASPP models have achieved higher DSC values than the Mask R-CNN models.

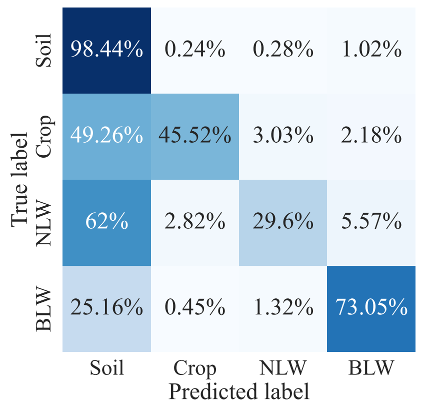
Table 4.3: Performance of the networks in classifying pixels belonging to the studied classes.

		Mask R-CNN		**Mask R-CNN-ASPP**		**U-Net-like**	
Class	**Metric**	**ResNet50**	**ResNet101**	**ResNet50**	**ResNet101**	**ResNet50**	**ResNet101**
Crop	DSC (%)	54.08	46.89	74.96	72.87	89.82	**92.29**
	IoU (%)	37.06	30.62	59.94	57.32	81.52	**85.67**
NLW	DSC (%)	37.59	22.24	57.34	58.00	85.42	**88.49**
	IoU (%)	23.15	12.51	40.19	40.85	74.55	**79.35**
BLW	DSC (%)	78.08	72.88	80.72	74.46	90.80	**92.24**
	IoU (%)	64.04	57.34	67.67	59.31	83.15	**85.60**
Soil	DSC (%)	97.39	96.49	96.62	96.01	98.57	**98.93**
	IoU (%)	94.91	93.22	93.47	92.34	97.18	**97.88**

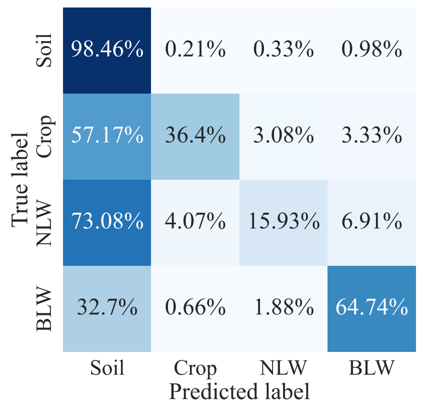
Specifically, the DSC of Mask R-CNN-ASPP-ResNet50, superior to Mask-R-CNN-ASPP-ResNet101, exhibited a superiority of 20.88 %, 19.75 %, and 2.64 % for the Crop, NLW, and BLW classes, respectively, compared to that obtained by Mask R-CNN-ResNet50, which surpassed Mask R-CNN-ResNet101. Nevertheless, U-Net-like-ResNet101, superior to U-Net-like-ResNet50, demonstrated a higher DSC by 17.33 %, 31.15 %, and 17.78 % for the Crop, NLW, and BLW classes, respectively, compared to that of Mask R-CNN-ASPP-ResNet50.

With respect to IoU, the achieved magnitude by Mask R-CNN-ResNet50 was superior to that obtained by Mask R-CNN-ResNet101 across all plant classes. However, Mask R-CNN-ResNet50 was surpassed by both Mask R-CNN-ASPP-ResNet50 and Mask R-CNN-ASPP-ResNet101 models. The IoU magnitude obtained by Mask R-CNN-ASPP-ResNet50 was 22.88 %, 17.04 %, and 3.63 % higher for the Crop, NLW, and BLW classes, respectively, than that exhibited by Mask R-CNN-ResNet50. Nevertheless, similar the DSC, the IoU of the U-Net-like models outperformed that of the Mask R-CNN-ASPP-based models for all plant classes. Specifically, the IoU obtained by U-Net-like-ResNet101 was 25.73 %, 39.16 %, and 17.93 % better for the Crop, NLW, and BLW classes, respectively, than that achieved by Mask R-CNN-ASPP-ResNet50.

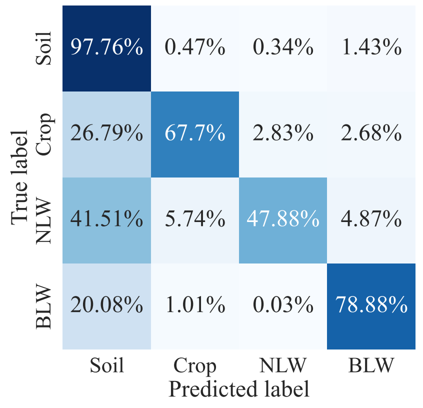
In summary, the Mask R-CNN-ASPP models exhibit superior performance compared to the Mask R-CNN models. However, the effectiveness of the Mask R-CNN-ASPP-based models is surpassed by the U-Net-like models.



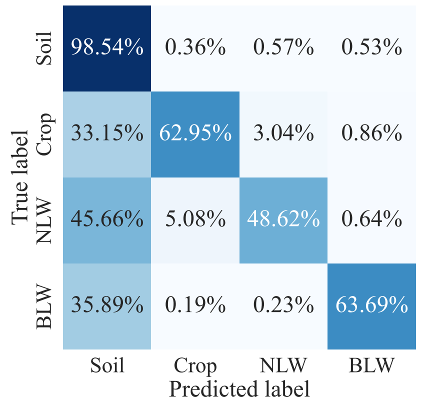
(a) Mask R-CNN-ResNet50



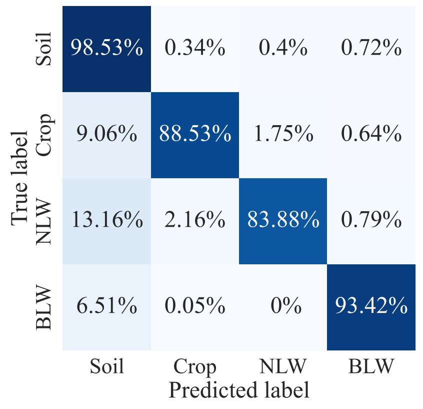
(b) Mask R-CNN-ResNet101



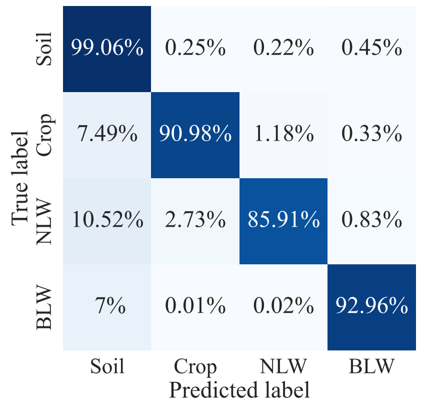
(c) Mask R-CNN-ASPP-ResNet50



(d) Mask R-CNN-ASPP-ResNet101



(e) U-Net-ResNet50



(f) U-Net-ResNet101

Figure 4.8: Confusion matrices depicting the pixel classification for both Mask R-CNN and U-Net configurations.

We have computed the confusion matrices presented in Figure 4.8 to demonstrate the accurate and misclassification of pixels in validation images for the classes Soil, Crop, NLW, and BLW for each model. All models achieved a classification rate of over 97 % for the soil class, which can be attributed to the predominance of soil in the images.

Examining the pixel classification of Corn, NLW, and BLW, all models performed best in classifying pixels belonging to the BLW class. In contrast, the least accurate classification was observed for pixels of the NLW class. The high accuracy in classifying BLW pixels can be attributed to the distinctive phenological appearance of the plant species within this group, making it clearly distinguishable from the plant species in the Crop and NLW classes. In contrast, the low classification accuracy of NLW pixels can also be attributed to the phenological characteristics of the plants in this group, as they are narrow and occupy a relatively small area in the images, often leading to misclassification as soil pixels.

Analyzing the pixel classification across model groups from Figure 4.8, Mask R-CNN-ResNet50 demonstrated superior classification of pixels for the three plant classes compared to Mask R-CNN-ResNet101. Conversely, the Mask R-CNN-ASPP-ResNet50 model exhibited better classification of pixels for the Crop and BLW classes than Mask R-CNN-ASPP-ResNet101, but not for the NLW class. In the case of U-Net-like networks, U-Net-like-ResNet50 effectively classified pixels for the BLW class. However, U-Net-like-ResNet101 achieved better classification for the Crop and NLW classes.

Comparing the top-performing model from each of the three groups in terms of pixel classification, it is clear that Mask R-CNN-ASPP-ResNet50 exhibited a 5.83 % higher accuracy in recognizing BLW pixels compared to Mask R-CNN-ResNet50. However, U-Net-like-ResNet50 outperformed by 14.54 % in classifying BLW pixels compared to Mask R-CNN-ASPP-ResNet50. Mask R-CNN-ASPP-ResNet50 demonstrated a 22.18 % superiority in recognizing Crop pixels compared to Mask R-CNN-ResNet50. Nevertheless, U-Net-like-ResNet101 surpassed the recognition of Crop pixels by 23.28 % compared to Mask R-CNN-ASPP-ResNet50. Lastly, U-Net-like-ResNet101 achieved a 56.31 % and 37.29 % better classification for the NLW class compared to Mask R-CNN-ResNet50 and Mask R-CNN-ASPP-ResNet101, respectively

In summary, as depicted in Figure 4.8, it is obvious that all models misclassified pixels of plant classes as soil. Additionally, there was a consistent confusion between pixels of the Crop and NLW classes in all models, and vice versa; similar to the classification model covered in previous Section 4.1.1.2. This behavior is attributed to the phenological characteristics of the plants, as they are monocotyledonous, potentially sharing certain features that lead to classification confusion.

Average performance of the models

The Figure 4.9 depicts the average performance of the Mask R-CNN, Mask R-CNN-ASPP and U-Net-like based configurations.

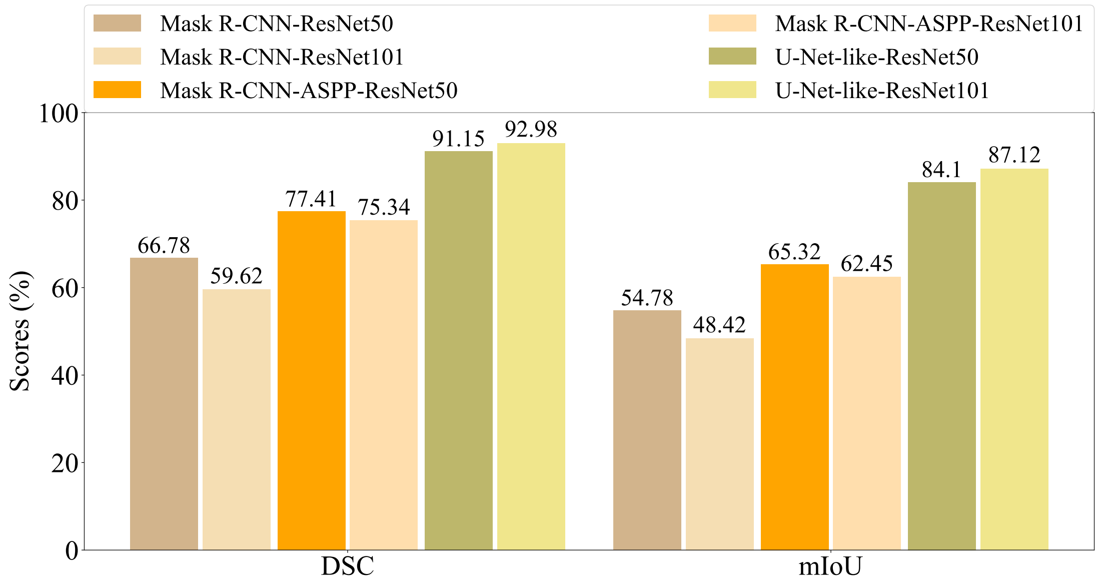


Figure 4.9: The average performance of the pixel-wise segmentation network configurations.

The U-Net-like networks surpassed both Mask R-CNN-ASPP-based and Mask R-CNN networks in semantic segmentation of the Soil, Crop, NLW, and BLW classes in our dataset. However, the two Mask R-CNN-ASPP-based models outperformed the Mask R-CNN-based models. Nevertheless, U-Net-ResNet101 achieved the highest values of the metrics. Concerning the performance of Mask R-CNN-based models, Mask R-CNN-ResNet50 demonstrated superior performance compared to Mask R-CNN-ResNet101, which exhibited the lowest performance.

Finally, Mask R-CNN-ASPP-ResNet50 demonstrated better segmentation of our dataset compared to Mask R-CNN-ASPP-ResNet101, as indicated by the metric values.

In terms of DSC, Mask R-CNN-ASPP-ResNet50 exceeded the achieved value of Mask R-CNN-ResNet50 by 10.63 %. However, the DSC of U-Net-like-ResNet101, the top-performing network model, was 15.37 % higher than that of Mask R-CNN-ASPP-ResNet50. Finally, the mIoU of U-Net-like-ResNet101 surpassed that of Mask R-CNN-ResNet50 and Mask R-CNN-ASPP-ResNet50, the best in their categories, by 32.34 % and 31.8 %, respectively.

Visualization of segmented classes

Visualizing the segmentation output of any model enhances the understanding of the numerical metrics. Hence, Figure 4.10 provides a qualitative comparison of the segmentation output from Mask R-CNN-ResNet50, Mask R-CNN-ASPP-ResNet50, and U-Net-like-ResNet101, which were the network architectures yielding the best results.

In the initial row, the input image is displayed (Figure 4.10a). The subsequent row (Figure 4.10b) depicts the ground truth, with green representing the Corn class, red for the NLW class, and blue for the BLW class. Following that, the third row (Figure 4.10c) showcases the segmentation output of the Mask R-CNN-ResNet50 model, while the fourth row (Figure 4.10d) exhibits the segmentation results produced by the Mask R-CNN-ASPP-ResNet50 model. Finally, in the last row (Figure 4.10d), the segmentation output of the U-Net-like-ResNet101 model is presented.

All three models exhibit accurate segmentation when the plants are well-separated, and the objects in the image are sufficiently large. These scenarios are expected when the image is captured from a short distance, as illustrated in the first column of Figure 4.10. Analyzing the segmentation performance of Mask R-CNN-ResNet50 and Mask R-CNN-ASPP-ResNet50, it becomes apparent that both models face challenges when there are more than two plant classes, when plants are in close nearby, and when plants appear small in the images, as observed in the second, third, and fourth columns of Figure 4.10c and Figure 4.10d. Additionally, these images reveal instances where Mask R-CNN-ResNet50 and

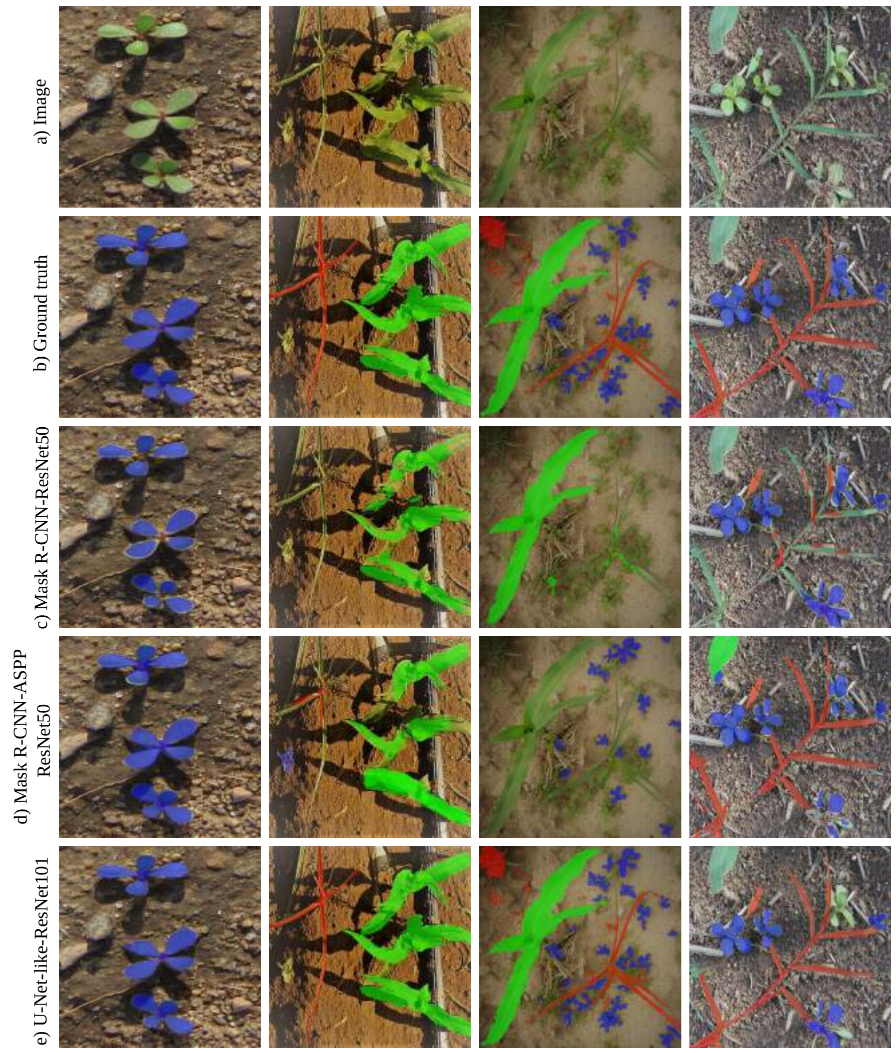


Figure 4.10: A visual assessment comparing the segmentation outputs of the top three models from each architectural configuration. The segmented crop plant is highlighted in green, NLW are shown in red, and BLW are represented in blue.

Mask R-CNN-ASPP-ResNet50 models commonly misclassify pixels belonging to the NLW class as the Soil class. However, in the image of the fourth column from the Mask R-CNN-ASPP-ResNet50 model, the NLW class has been accurately segmented, attributed to the ASPP module implemented in its segmentation branch. This mosaic of images underscores that the U-Net-like-ResNet101 model outperforms in segmenting all classes, with its segmentation output masks closely aligning with the ground truth under these real field conditions.

4.1.1.4. Segmentation and classification for detection

The performance of the networks involved in the proposed approach, combining segmentation and classification CNNs for plant detection, is presented in this section. First, the segmentation results carried out by the U-Net-like-ResNet101 architecture are presented. Furthermore, the achievements of the classification networks ResNet101, VGG16, Xception, and MobileNetV2 are also showcased, as they were evaluated in classifying the segmented regions. Finally, a collection of representative output images from the detection system is presented.

Performance of the U-Net-like model for segmentation

The segmentation stage has been conducted using two approaches. The first involves segmenting the entire resized input images, while the second approach entails dividing the input images into patches, followed by segmentation of each patch. Five experiments were conducted using both the NIR images (Figure 3.9) and the visible spectrum dataset (Figure 3.10).

Table 4.4: Performance of the U-Net-like-ResNet101 network when trained with RGB and multispectral images.

		Resized image				Patched image
Class	**Metric**	**Experiment 1**	**Experiment 2**	**Experiment 3**	**Experiment 4**	**Experiment 5**
Crop	DSC (%)	83.97	18.18	60.83	5.65	**86.72**
	IoU (%)	72.36	8.21	43.71	2.91	**77.18**
NLW	DSC (%)	68.51	2.65	49.48	15.29	**73.41**
	IoU (%)	52.1	1.34	32.87	8.28	**58.03**
BLW	DSC (%)	86.68	0.51	68.41	34.79	**91.64**
	IoU (%)	76.69	0.26	55.29	21.06	**81.14**
Soil	DSC (%)	97.78	91.06	96.15	93.70	**98.16**
	IoU (%)	95.66	83.59	92.58	88.14	**96.32**

Experiment 1: U-Net-like-ResNet101 trained with RGB images and tested in RGB images. **Experiment 2:** U-Net-like-ResNet101 trained with NIR images and tested in RGB images. **Experiment 3:** U-Net-like-ResNet101 trained with RGB images, transferring encoder weights acquired during NIR training, and tested in RGB images. **Experiment 4:** U-Net-like-ResNet101 trained with NIR and RGB images and tested in RGB images. **Experiment 5:** U-Net-like-ResNet101 trained with RGB images and tested in RGB images when images were of original size and divided in patches.

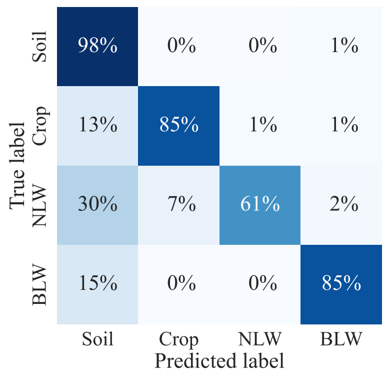
In this way, Table 4.4, in each column, lists the performance of the model for the five carried-out experiments. Observations reveal that the U-Net-like-ResNet101 model achieved better segmentation of the four classes in the dataset when images were divided into patches (experiment 5). The second better model was obtained when trained exclusively with resized RGB images (experiment 1). Fine-tuning the encoder weights with pure NIR images and subsequently retraining the decoder using only RGB images (experiment 3), was also acceptable but it did not achieved the performance obtained in experiment one and five. However, combining NIR and RGB images for training did not yield improvement in segmentation, as indicated by the metrics (experiment 4). Notably, training the model solely with NIR images resulted in the poorest segmentation performance (experiment 2).

Despite the superiority of training U-Net-like-ResNet101 with patched images compared to other dataset configurations, its performance experienced a decline in this experiment compared to that observed in Section 4.1.1.3. This decline can be attributed to the new dataset's size, which includes unknown plant species in NLW and BLW groups, introducing a variety of features in these two classes.

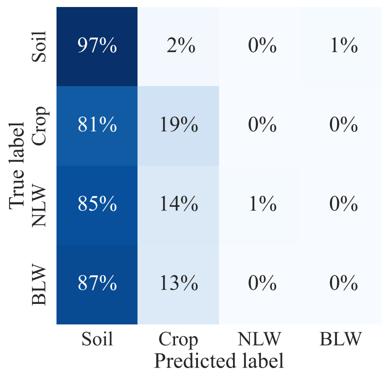
Additionally, Table 4.4 indicates that the two better models performed more effectively in segmenting the BLW class, followed by the Crop class, while the NLW class exhibited the poorest segmentation. A closer analysis highlights that, when images were divided into patches, the classes Crop, NLW, and BLW demonstrated segmentation improvements of 2.75 %, 4.90 %, and 4.96 %, respectively for the DSC, compared to when resized RGB images were used for training.

The U-Net-like model exhibits a consistent behavior for the IoU metric across both experiment one and experiment five. Specifically, the model performs more effectively for all classes when images are divided into patches. The IoU analysis reaffirms that the BLW class achieved the best segmentation, followed by the Crop class, with the NLW class being the least effectively segmented. Segmenting using patches increased 4.82 %, 5.93 %, and 4.45 % in the IoU metric for the Crop, NLW, and BLW classes, respectively, compared to the IoU values obtained when RGB images were solely resized.

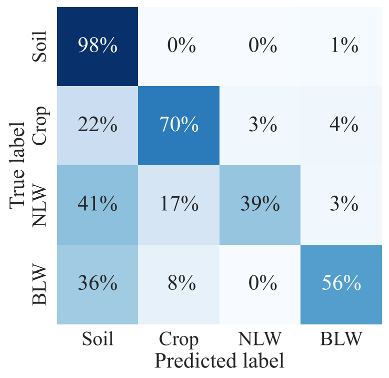
The percentage of pixels correctly classified or misclassified by the U-Net-like model under the five experiments is illustrated in Figure 4.11 through confusion matrices. In all cases, the model demonstrated improved pixel classification for



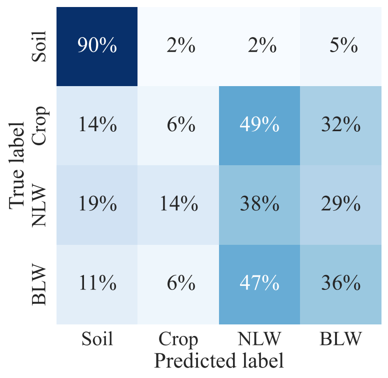
(a) RGB



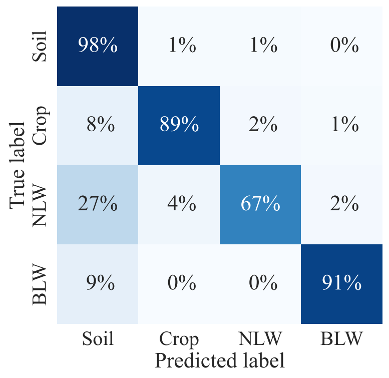
(b) NIR



(c) Transfer NIR-RGB



(d) NIR-RGB



(e) Patches

Figure 4.11: Confusion matrices for U-Net-like-ResNet101 trained under distinct datasets configuration. (a) Architecture trained with RGB images and tested in RGB images, (b) architecture trained with NIR images and tested in RGB images, (c) network trained with RGB images, transferring encoder weights acquired during NIR training, and tested in RGB images, (d) architecture trained with NIR and RGB images and tested in RGB images, and (e) network trained with RGB images and tested in RGB images when images were of original size and divided in patches.

plant classes when images were divided into patches compared to the sole resizing the RGB images, which was the second better model. The inferior model, as indicated in Figure 4.11(b), was the model trained exclusively with NIR images and tested on RGB images. Under the two superior segmentation approaches, the classes Crop and BLW were better segmented than the class NLW. Additionally, it is noteworthy that, under these two scenarios, the model tended to misclassify pixels belonging to plant classes as soil to a greater magnitude.

The overall performance of the U-Net-like-ResNet101 architecture under the five experiments is presented in Figure 4.12. The graph illustrates that the best-performing model is the one trained with patched images, applying transfer learning for the ResNet101 backbone from ImageNet. The second-best model is achieved when resized RGB images were used for training. Conversely, the worst-performing model is the one trained using NIR images and evaluated on RGB images. The differences in DSC, and mIoU metrics between the model trained with patched images and the model trained with resized RGB images are quantified as 3.21 %, and 3.96 %, respectively.

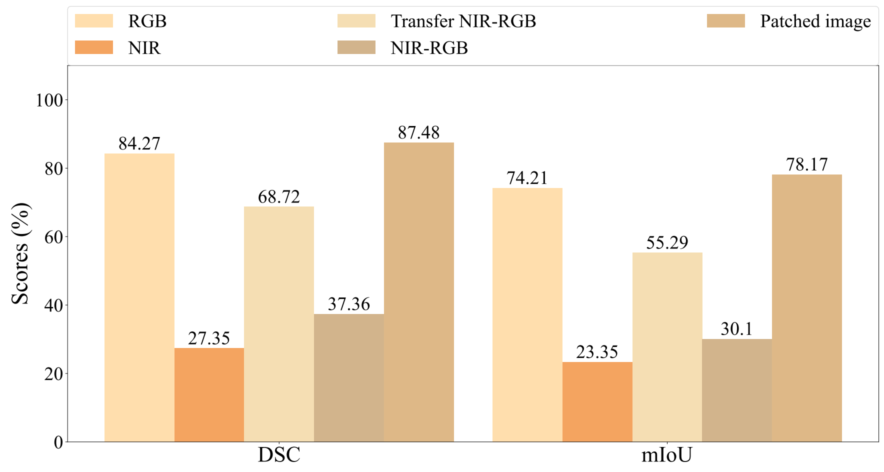


Figure 4.12: Average performance of U-Net-like-ResNet101 under the different configurations of the datasets and transfer learning strategies.

Performance of the classification networks

The fine-tuning process revealed that applying transfer learning led to improved classification performance for the networks. Consequently, only the weights of the FCL were adjusted to suit our dataset. The fully connected block consisted of three layers, and it was observed that superior classification results were achieved when both the first and second layers were equipped with 2,048 neurons. Additionally, during training, the Adam optimizer was employed with a learning rate set to 0.0001. The use of the categorical cross-entropy loss function contributed significantly to error reduction. The training process involved 50 epochs on the entire dataset.

The macro performance of ResNet101, VGG16, Xception, and MobileNetV2 in classifying ROIs extracted from segmented images is illustrated in Figure 4.13. As observed, Xception exhibited the highest performance, followed by MobileNetV2, ResNet101, and VGG16, as indicated by the metrics of Accuracy, Precision, Recall, and F1-score.

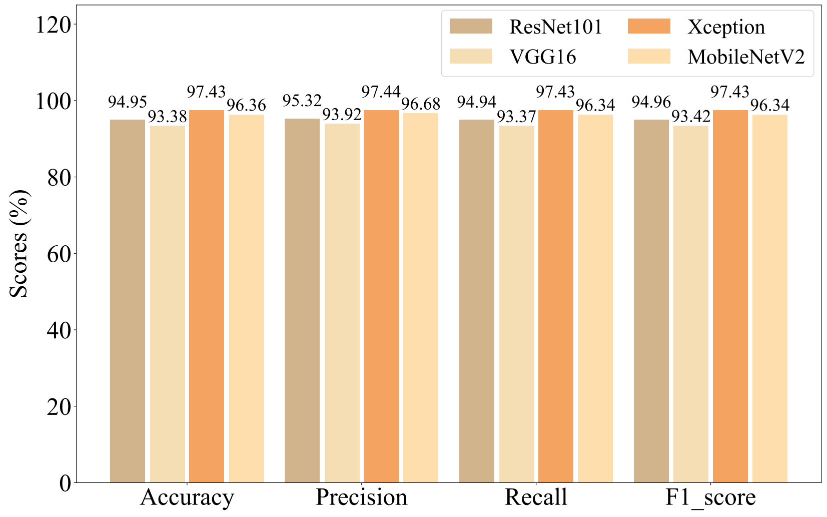


Figure 4.13: Average performance of the studied classification networks.

In real-world applications, inference time plays a crucial role. Among the studied classification networks, the computation cost of the MobileNetV2 network stands out, being 8 to 9 times smaller than the other architectures. This efficiency is attributed to its implementation of depthwise separable convolution, which combines depthwise convolutions and pointwise convolutions, resulting in a reduction of trainable parameters (Sandler et al., 2018).

Figure 4.14 highlights the performance of the MobileNetV2 model in classifying plants belonging to the Crop, NLW, and BLW classes. Analyzing the Recall metric reveals that 100 % of images from the Crop class were correctly classified, along with 90 % for NLW and 99 % for BLW by the MobileNetV2 model. However, the Precision for the Crop class is 90 %, indicating a misclassification rate of 10 % of NLW plants as Crop. This is due to the 100 % Precision for the NLW class, implying that the model misclassifies some NLW plants as Crops. Consequently, examining Precision, Recall, and F1 score for BLW, makes us realize that this class is the best classification. Finally, the mean classification performance across classes was 95 %, 95 %, and 99 %, for Crop, NLW, and BLW, accordingly, which is indicated by the F1-score.

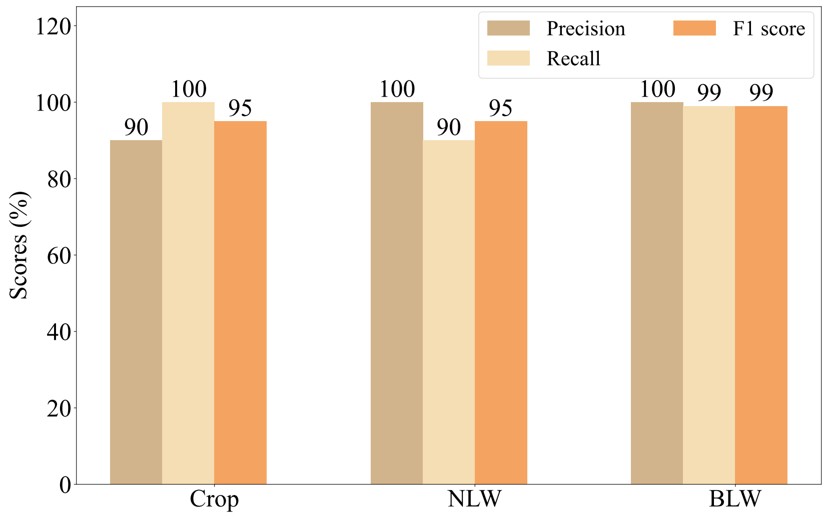


Figure 4.14: MobileNetV2 classification performance over the plant classes of our dataset.

Detection approach visualization

Identifying objects within an image and determining their corresponding class is the essence of object detection. Figure 4.15 illustrates the detection of plant classes through our proposed approach, showcasing a variety of images. The initial two rows display images with sparse plant distribution and no foliage occlusion. In contrast, the subsequent two rows feature images with high plant density and persistent foliage occlusion. It is noteworthy that, in all samples, green boxes correspond to the Crop class, red boxes to NLW, and blue boxes to BLW.

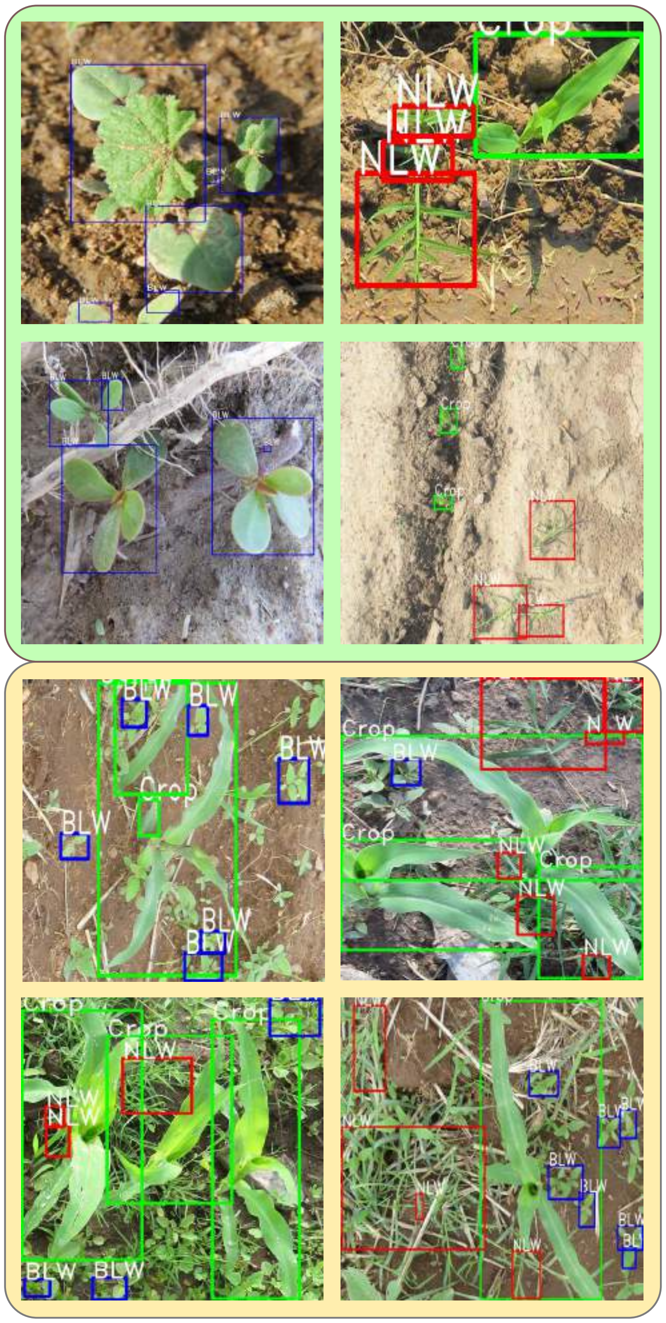


Figure 4.15: Samples of output images generated by the proposed detection method, utilizing both the segmentation and classification networks. The first two rows contain low-density plant images. The last two rows contain high-density plant images. In all samples, the green box is Crop, the red box is NLW, and the blue box is BLW.

A visual examination of images with a low plant density reveals that nearly all green regions have been successfully detected. However, the localization of plants is somewhat influenced by the region provided by the segmentation model, often resulting in multiple bounding boxes in a single image. In images with a high plant density, most plant classes are detected, but due to the dense foliage, bounding boxes sometimes encompass two or more plants of the same class, influenced by the region extracted by the segmentation model. Furthermore, in high-density plant images, some plants are not detected, primarily due to the segmentation model's confusion between pixels belonging to plant classes and those representing the soil.

Although the detection in some cases covers only part of the foliage of the plants, the implementation of this vision system for spraying herbicides in real corn fields remains suitable. This is because systemic herbicides are absorbed by the plants and gradually distributed throughout their vascular system, effectively killing all their organs. Thus, spraying herbicides over just a portion of the plants' foliage is sufficient for these herbicides to eliminate the plants. In instances where the trained segmentation model identifies multiple plants in a region, this can be addressed by subdividing the bounding box. This approach allows for the spraying of a smaller area of foliage, optimizing the efficiency of herbicide application.

4.1.1.5. End-to-end segmentation based on transformers

This section presents the achievements of the transformer architectures Swin-UNet, Segmenter, and SegFormer on pixel-wise segmentation of the Crop, NLW, BLW, and Soil classes of our dataset. The obtained metrics of the architectures over each class are listed in Table 4.5. Upon examining this table comprehensively, it becomes apparent that SegFormer stands out as the superior transformer model for every class. The second better was Segmenter and the worse performance was reached by Swin-UNet.

Focusing solely on the plant classes, SegFormer demonstrated superiority for the metric DSC. Specifically, it outperformed the Segmenter and Swin-UNet models by 17.98 % and 75.64 %, respectively, for the class Crop; by 44.81 % and 63.54 %, respectively, for the class NLW; and by 12.55 % and 19.69 %, respectively, for the class BLW.

Table 4.5: Performance of the transformers architectures on segmenting the classes of the dataset.

Metric	Classes	Swin-UNet	Segmenter	SegFormer
DSC (%)	Soil	96.67	97.89	**98.36**
	Crop	15.68	73.34	**91.32**
	NLW	15.74	34.47	**79.28**
	BLW	70.33	79.45	**90.00**
IoU (%)	Soil	93.55	95.88	**96.77**
	Crop	8.51	57.91	**84.02**
	NLW	8.54	20.82	**65.67**
	BLW	54.24	65.91	**85.18**

Regarding the metric IoU, SegFormer surpassed Segmenter by 26.11 %, 44.85 %, and 19.27 % for the classes Crop, NLW, and BLW. Similarly, for the same metric, SegFormer outperformed Swin-UNet by 75.51 %, 57.13 %, and 30.94 % for the classes Crop, NLW, and BLW, respectively.

The metrics DSC and IoU are closely associated with the quality of segmentation performed by the models. Table 4.5 facilitates a comparison of model performance across plant classes in the dataset. In this way, all models obtained a superior magnitude of these metrics for the class BLW. Furthermore, it seems that SegFormer and Segmenter excel in segmenting the Crop class compared to the NLW class, as indicated by both IoU and DSC. In contrast, Swin-UNet appears to achieve similar segmentation results for the NLW and Corn classes, according to these metrics. The improved segmentation of the BLW class by all three models may be attributed to the distinct leaf appearance of the plant species within this class, which is entirely different from both Crop and species in the NLW class. Additionally, BLW plants typically occupy substantial spatial areas.

The confusion matrices resulting from the evaluation of the transformers on the testing dataset are depicted in Figure 4.16. These matrices provide insights into the performance of correct classification and misclassification of pixels belonging to each class.

Analyzing Figure 4.16(a) with a focus on plant classes, Swin-UNet demonstrated superior classification of BLW class pixels compared to the NLW class, while the worst performance was observed in classifying Crop pixels. Notably, this model exhibited significant confusion, misclassifying a considerable percentage of BLW pixels as NLW and Crop. Additionally, Swin-UNet showed a misclassification of NLW pixels, incorrectly identifying them as Crop. Moreover, the model misclassified almost all pixels belonging to the Crop class as if they were other classes.

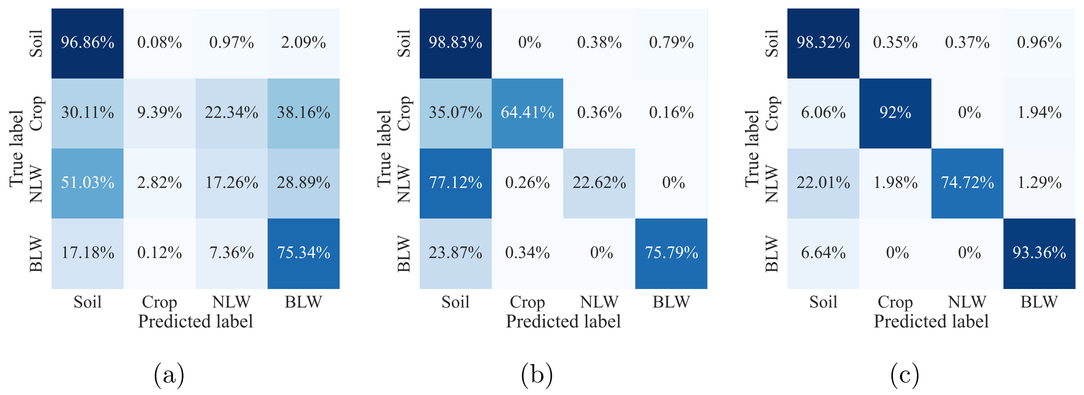


Figure 4.16: Confusion matrices of the three studied transformer architectures. (a) Swin-UNet, (b) Segmenter and (c) SegFormer.

On the other hand, regarding the confusion matrix of the Segmenter transformer (Figure 4.16(b)), it also exhibited a high rate of classification accuracy for pixels belonging to the BLW class, followed by those of the Crop class, and finally the pixels of the NLW class. In contrast to Swin-UNet, this model did not exhibit confusion among pixels of different plant classes; instead, it tended to misclassify them as soil pixels.

Finally, SegFormer, being the superior model, demonstrated the most accurate classification for pixels belonging to the BLW class, followed by pixels of the Crop class. The least accurately classified were the pixels of the NLW class, as illustrated in Figure 4.16(c). Similar to Segmenter, these models exhibited a higher degree of confusion, misclassifying pixels of plant classes as if they belonged to the soil class.

In general, Figure 4.17 indicates the macro performance of the evaluated transformers with respect to DSC and mIoU. The graph reaffirms the superior performance of SegFormer, achieving the highest metrics. Subsequently, Segmenter and Swin-UNet follow. The overall DSC obtained by SegFormer was 19.95 % and 40.64 % higher than that achieved by Segmenter and Swin-UNet, respectively. Finally, in terms of mIoU, SegFormer surpassed Segmenter by 22.78 % and Swin-UNet by 41.7 %.

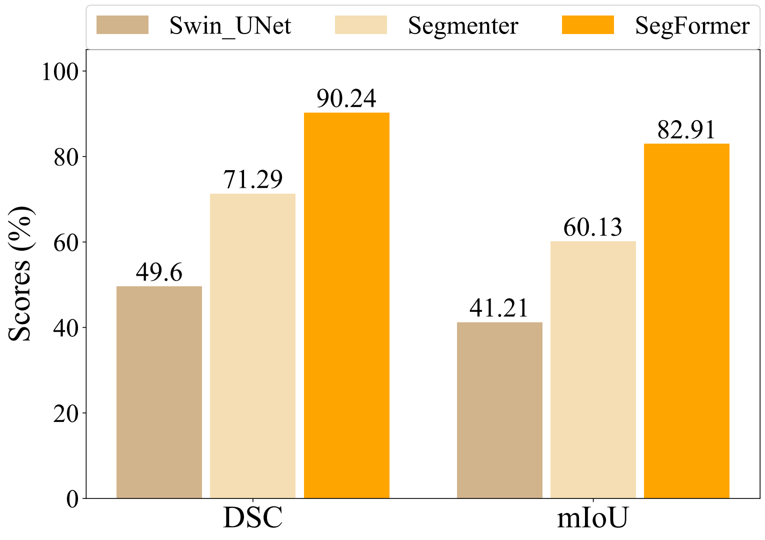


Figure 4.17: Average performance of the three studied transformers architectures.

Visualization of segmentation

A sample of images pixel-wise segmented by the trained transformer SegFormer is shown in Figure 4.18. This architecture demonstrated superior performance on metrics Precision, DSC, and IoU for the segmentation of Crop, NLW, and BLW. The first column (Figure 4.18(a)) represents the input image capture, while the second column (Figure 4.18(b)) depicts the ground truth. The last column displays the prediction image (Figure 4.18(c)).

From this set of sample images, the first three rows depict images captured at a medium distance from the soil surface to the camera, while the last three rows showcase images captured at a greater distance from the soil surface to the camera. In this way, in the first and second rows, the segmentation of the classes Crop and BLW is presented. It is observed that almost all pixels of the Crop class were correctly classified. However, some pixels of the BLW class were misclassified as Soil. Despite this minor misclassification, the model shows an inclination to

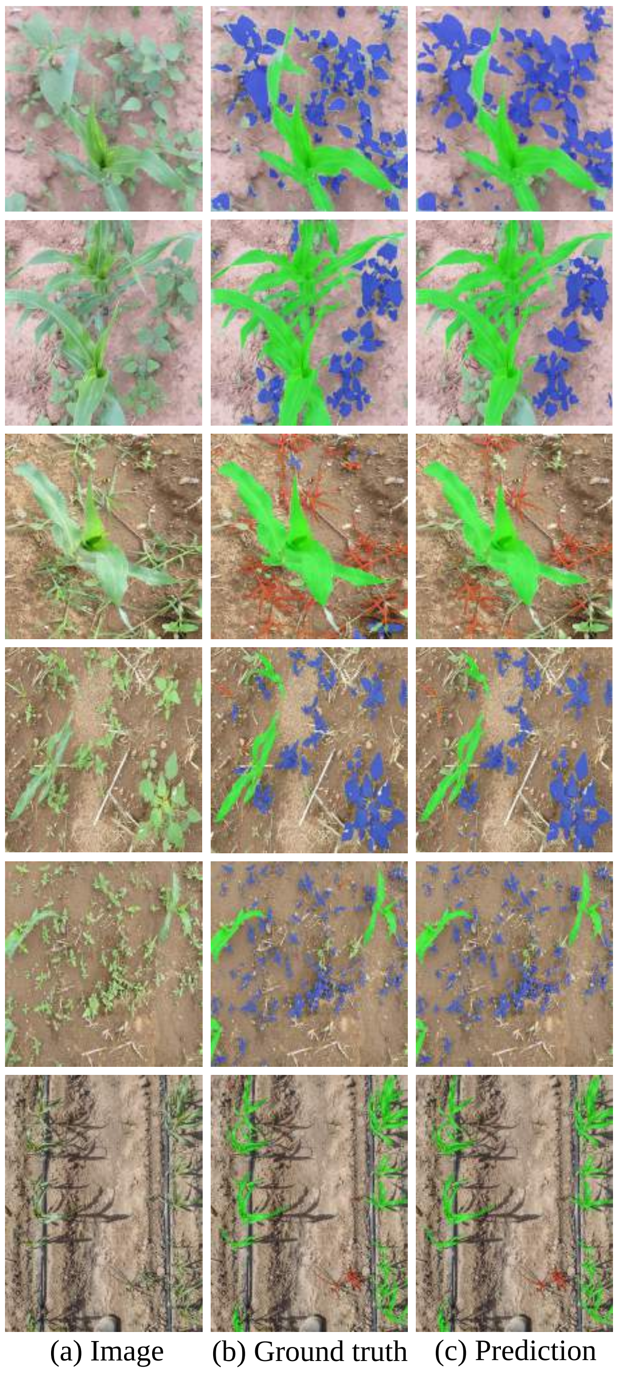


Figure 4.18: Samples of pixel-wise segmentation performed by SegFormer transformer of the classes Crop, NLW, and BLW under natural cornfields. Green, red, and blue ROIs belong to the classes Crop, NLW, and BLW, correspondingly.

enhance the segmentation of BLW ROIs by tracing a more defined profile over the plants compared to the ground truth. The Corn and NLW classes in the third row were correctly isolated, nonetheless, the pixels of the class BLW were misclassified as Soil.

The pixels corresponding to the plants in the last three rows were accurately classified, as the prediction closely resembles the ground truth. This observation reinforces the model's tendency to better recognize pixels belonging to the class BLW, as evidenced by the isolation of nearly all pixels associated with small plants.

4.1.2. Mechatronic platform

This section outlines the design of the three primary systems of the mechatronic platform, referred to as the SWS. These systems encompass mechanical design, electrical and hydraulic design, and vision release. However, the election of the better concept, considering the problem and restrictions is first addressed.

4.1.2.1. Conceptual design: alternative election

Considering the platform should be experimental and low-cost, the elected alternatives from the morphological chart for the mechanical, electrical and hydraulic, and vision systems are listed in Table 4.6. As observed, we chose to develop an assisted navigation platform pulled by an agricultural tractor. This alternative aligns with the main thesis objective, reduces implementation time and fabrication costs, and could be a viable option for farmer adoption in a short time.

Taking into account soil irregularity and the tractor engine, the platform would be subjected to constant vibrations. Therefore, for the mechanical elements, it was decided to use steel profiles to fabricate the main chassis. The chassis components would be joined by welding, and some other components would be screwed into each other. Concerning chemical containers, we opted to use plastic containers because herbicides are corrosive.

Concerning the electrical and hydraulic system, electrical solenoid valves were chosen to control the herbicide flow to the sprayer nozzles. This type of valve has

Table 4.6: Elected alternatives for the systems of the SWS.

	Alternatives			
Systems and sub-systems	**1**	**2**	**3**	**4**
Navegation				
Type	Autonomous	[✓ Pulled by tractor]	–	–
Mechanical design				
Material	Wood	Aluminium	[✓ Steel]	Plastic
Assembly	[✓ Welded]	[✓ Screwed]	–	–
Chemical tank	[✓ Plastic]	–	–	–
Electrical and hydraulic design				
Power supply	Gasoline generator	[✓ LiPo battery]	Supplied from tractor	[✓ Car battery]
Controller	[✓ Arduino]	PIC	Raspberry Pi	–
Solenoid valve	Hydraulic activation	Pneumatic activation	[✓ Electric activation]	
Nozless	Flat fan	[✓ Solid cone]	Hallow cone	Adjustable
Pump	Diaphragm (Mechanical)	[✓ Diaphragm (Electrical)]	–	–
Vision system release				
Processing type	[✓ *insitu*]	Remote		
Camera type	Smartphone	Professional	[✓ Webcam]	

a response time of 30 ms, which is suitable for the vision system. Additionally, a solid cone nozzle was selected due to its cost-effectiveness and adaptability to the task. Additionally, Arduino Mega was elected as the controller to relay the electrical signal to the actuators. For sending the chemical mixture from the tanks to the nozzles, it was decided to use electrical diaphragm pumps. The system was powered by a LiPo battery and a Car battery to ensure a stable voltage and to prevent current spikes.

Finally, for the vision system release, the decision was made to capture the images from the actual field with a webcam and subsequently process them *in situ* using a compact computer.

The following Table 4.7 lists the main components and their technical features used for the fabrication and instrumentation of the mechatronic platform.

Table 4.7: List of main components that integrate the SWS.

Materials	Quantity	Features	Image
AISI 1013 Carbon Steel Profiles	–	Density: $7.7 - 8.3 kg\, m^{-3}$ Poison's Ratio: $0.27 - 0.30$ Elastic modulus: $190 - 210 GPa$ Yield Strength: $1034 MPa$	
Herbicide tanks	2	Volumen: $38L$	
LiPo battery	1	Tension: 24VDC Current: $17,000\, mAh$ Cells: 6	
Car battery	1	Model: L-58-575 Tension: 12VDC	
Arduino Mega 2560	1	Microcontroller: ATmega2560 Input voltage: 7-12 VDC Digital input/output pins: 54 Clock speed: 16MHz	
Pump	2	Model: – Input voltage: 12VDC Power: $60W$ Max flow: $5L\, min^{-1}$	

Table 4.7: Continue.

Materials	Quantity	Features	Image
Power Inverter	1	Model: – Input voltage: 12-24 VDC Output voltage: 110 VAC Max power: 750 W	
Nozzle	12	Model: VP-110-01 Pressure: $0.5MPa$ FLow rate: $0.52L\,min^{-1}$ Connector: 8mm	
Solenoid valve	12	Model: 2w-040-10 Input voltage: 12VDC Orifice: 4 mm Temp: $-5 - 80C^{\circ}$ Pipe size: $\frac{3}{8}in$ Max pressure: $980.6kPa$	
Touch screen	1	Model: 7inch HDMI LCD (C) Power supply: 5VDC Screen size: 7 in Resolution: 1024 × 600 Display interface: HDMI	
Relay module	1	Model: SRD-12VDC-SL-C Modules: 16 Input voltage: 12VDC Coil voltage: 12VDC Operating current: 10A Activation current: 15-20 mA	
NVIDIA Jetson AGX Xavier	1	GPU: 512-core NVIDIA Volta with 64 Tensor Cores GPU Max Frequency: 1377 MHz CPU: 8-Core ARM v8.2 64-Bit Carmel Memory: 32GB 256-Bit LPDDR4x—136.5GB/s	

Table 4.7: Continue.

Materials	Quantity	Features	Image
Webcam	1	Model: ELP-USBFHD01M-SFV Pixel size: $12.8 \times 11.6mm$ Image size: 2.4Mp Frames: 30fps Lens parameters: 2.8-12mm CS Connecting port: USB 2.0 Power supply: USB 5VDC	

4.1.2.2. Detailed design of the smart weed sprayer

Mechanical system

The conceptualized idea was drawn in SolidWorks, whose result is shown in Figure 4.19. As observed, the platform is composed of a main chassis with a three-point hitch for its coupling to tractors. This hitch was standardized for category II tractors, according to the norm ASAE S217.12 (ASABE, 2006), because they are the most common in Mexico. The chassis' main dimensions such as large, width, and height are $1.85m$, $0.80m$, and $1.30m$, correspondingly. As observed, the chassis supports all the elements and actuators, such as a camera, herbicide tanks, nozzles, solenoid valves, pumps, and also, this structure includes a protection case for the electrical component of the systems.

The chassis includes a versatile frame for the camera that allows it to adjust its vertical position, getting closer or moving away from the soil surface at intervals of $10cm$. Furthermore, the camera can be horizontally moved, either near or far from the nozzles, with increments of $8cm$. Also, the frame allows the camera to rotate freely, as shown in Figure 4.19(b). This feature is quite important for calibrating the captures in field practice.

As observed, the SWS was thought to contain two herbicide tanks. One of them could be used to hold herbicides specially developed to control NLW and the other to hold herbicides developed to control BLW. However, in case a broad-spectrum herbicide is used, capable of controlling NLW and BLW, could be used solely one of the tanks. Therefore, a sprinkler rail with six nozzles was included in the prototype for each herbicide tank. The sprinkler rails are separated by 11cm from each other, and the nozzles are 20cm apart from each other.

Electrical and hydraulic system

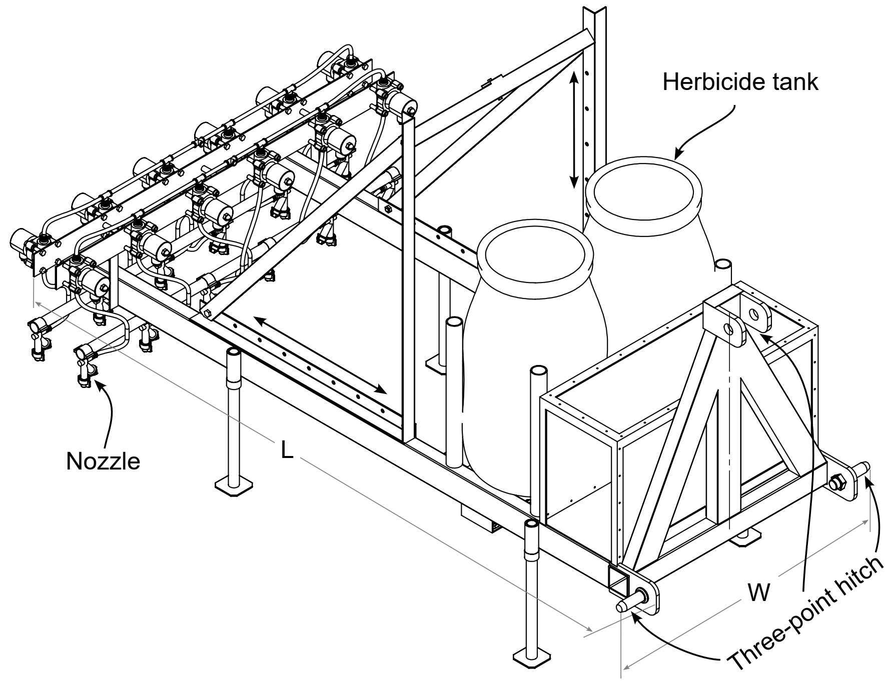
The electric and hydraulic diagram of the SWS is shown in Figure 4.20. The blue lines represent the hydraulic ducts for conducting the herbicides, whereas the electrical connections are represented in black. Specifically, the energy signal is shown in black dotted lines.

As aforementioned, the prototype includes two herbicide tanks; therefore, each of them is connected to a pump to send the herbicide mixture to the nozzles. After each pump, a relief valve was adapted to open at 689kPa. These relief valves are used because the system operates in an "on" or "off" mode. This means that the solenoid valves remain closed until the vision system detects weeds in the sprayer zone of the nozzles, and then they open. If no weeds are detected, the solenoid valves remain closed for an extended period, then the pressure could potentially damage the hydraulic ducts, the pump, and the solenoid valves.

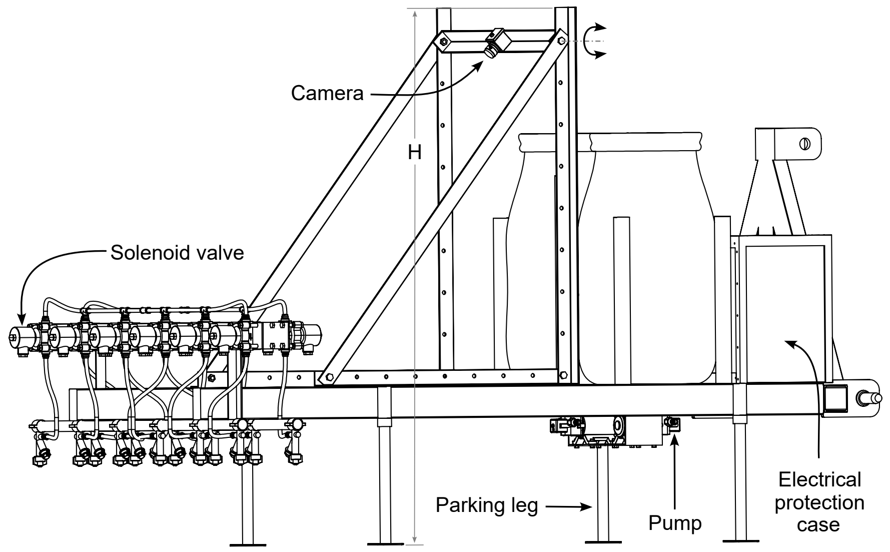
The principal electrical components include the NVIDIA PC, a camera, a screen, an Arduino Mega as the controller, two pumps, 12 solenoid valves, and a 16-relay module to actuate the solenoid valves and pumps according to signals. The basic diagram is also illustrated in Figure 4.20.

The 24VDC LiPo battery powered the NVIDIA PC, the screen, and the Arduino Mega. However, for practicality, a 24VDC to 127VAC inverter was used to plug these elements using its original cables. On the other hand, the 12VDC car battery fed the power electronic equipment, including the two pumps, the 12 solenoid valves, and the 16-relay module. The camera received power from the NVIDIA PC through its USB cable, and the Arduino Mega also provided a tension of 5VDC to the pins of the relay module.

From the 16-relay module, 14 of them were utilized to activate the 12 solenoid valves and the two pumps. Therefore, each of them was connected to an assigned digital output of the Arduino Mega. This setup allows the relay to be activated, opening the solenoid valve, in accordance with the vision detection algorithm, when a weed is detected in the spraying zone of the nozzles.



(a)



(b)

Figure 4.19: Isometric views of the mechatronic platform drawing. (a) front-view isometric and (b) back-view isometric.

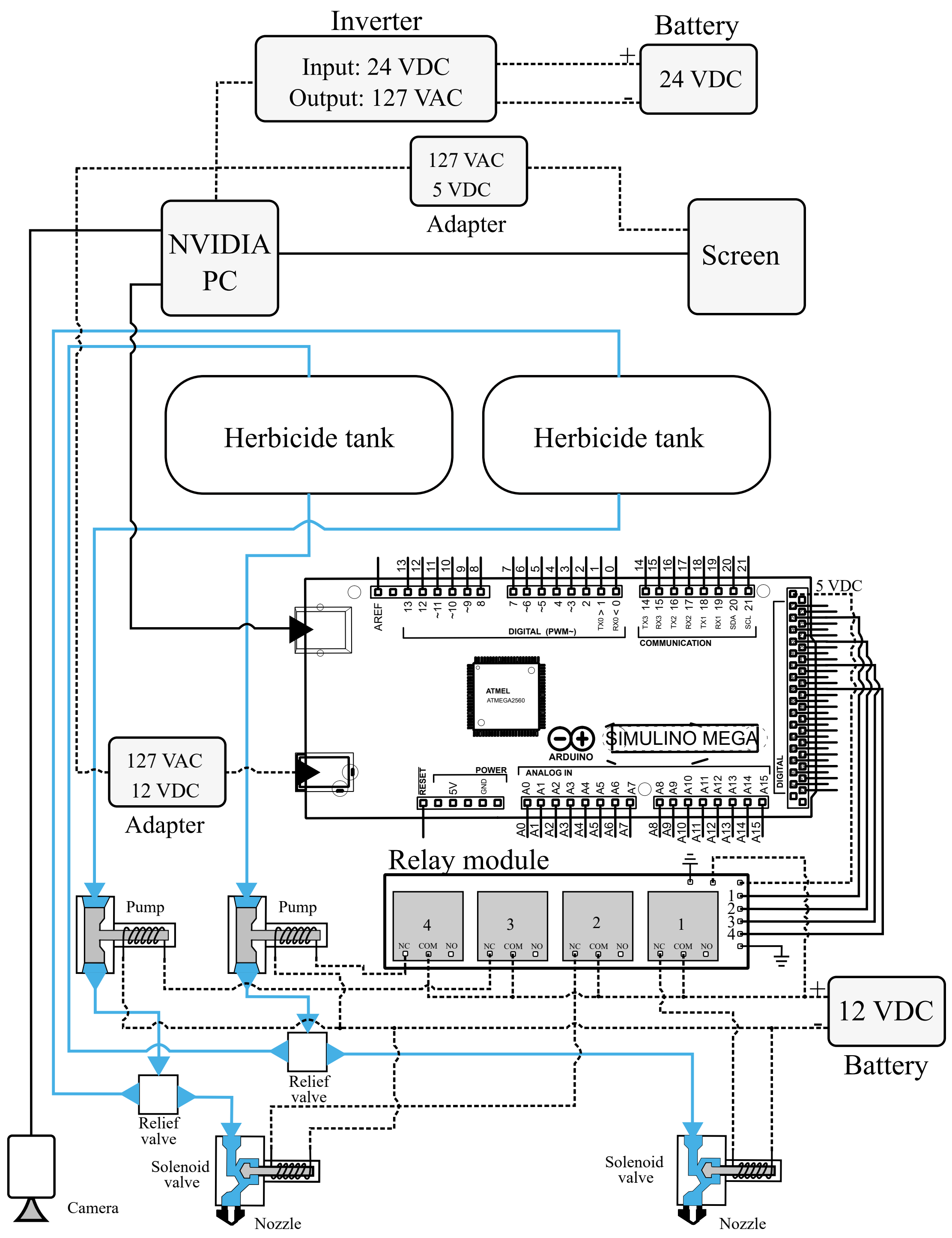


Figure 4.20: Electrical and hydraulic diagram of the smart sprayer.

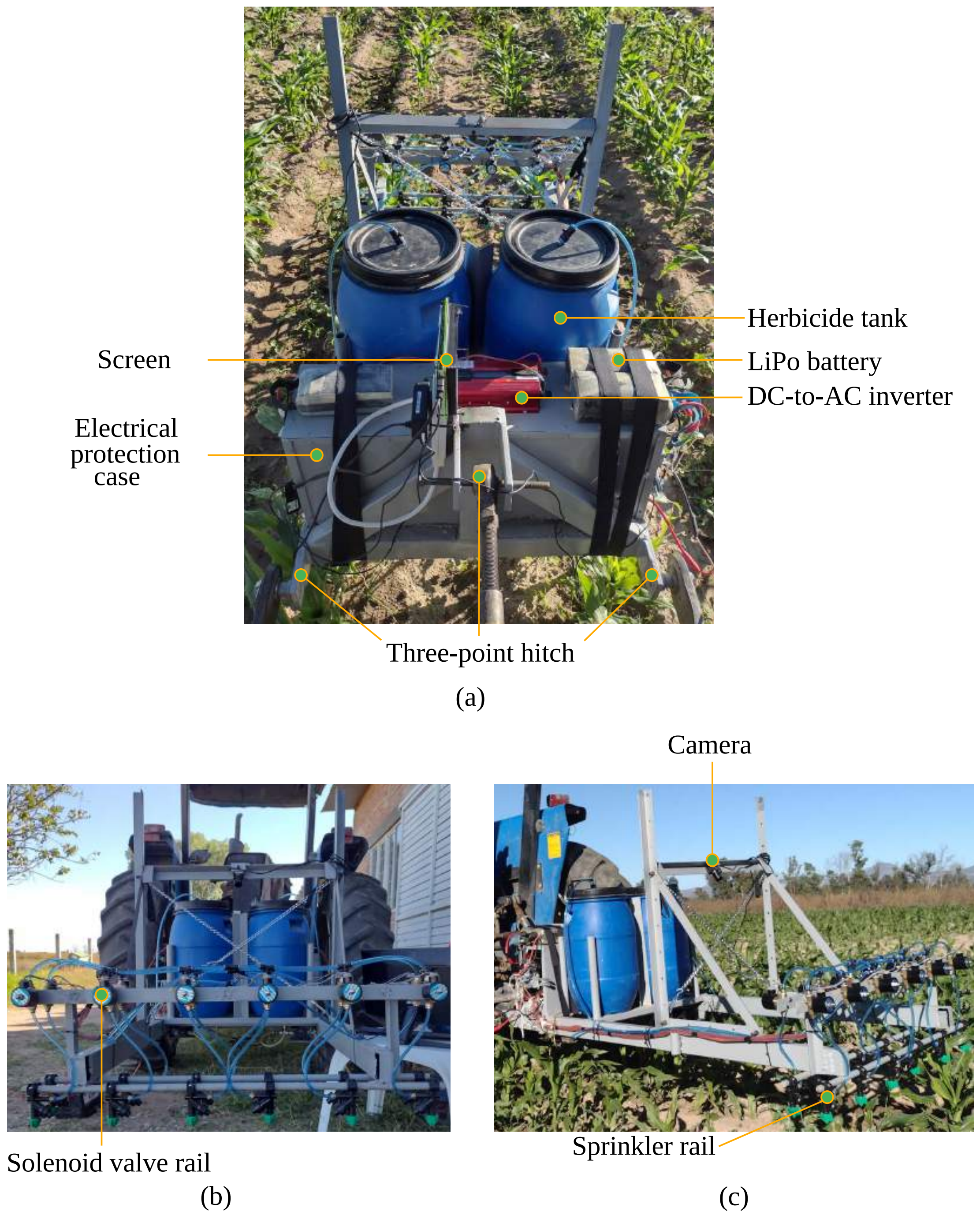


Figure 4.21: Smart weed sprayer prototype. (a) Front view isometric, (b) back view isometric, and (c) lateral view isometric.

After the drawing of the mechanical system and having designed the electrical and hydraulic systems, the mechatronic platform was constructed and instrumented. These tasks were completed prior to the implementation of the vision system system. The electrical system operates based on the instructions of the vision system; hence, it needed to be installed first in the mechanical structure to calibrate the vision system efficiently in field environments. Figure 4.21 shows three image samples, in which three views of the fabricated prototype are observed. Some components have been highlighted in these images.

Vision system release

Once a trained deep learning model was available for detecting weeds under natural cornfields and an instrumented mechatronic platform, the challenge was correlating the vision system with the localization of the nozzles, which we named vision calibration. To achieve this, the following four main actions were performed: (a) computation of the nozzles' coordinates spatially into the prediction image, (b) computation of a spray factor for every nozzle, which helps determine the moment each nozzle should spray, (c) divide the prediction image into stripes; since each stripe is the aspersion width of the nozzles, and (d) release of the vision system.

Actions (a), (b), and (c) for calibration are illustrated in Figure 4.22, with four stages. However, previously to perform the calibration, a calibration rug was fabricated with 12-circular red regions disposed equally to the nozzles arrays. Subsequently, the calibration rug is carefully placed under the nozzle arrays, aligning each circular region with the corresponding nozzle in the array. In stage one of this process, the camera of the SWS, fixed in the working position, captures an image of the calibration rug, covering the region of interest in front of the nozzles.

Afterward, in stage two the calibration image is thresholded and binarized according to the methodology aforementioned in Section 3.1.1.2. The thresholding values were set manually until the correct isolation of the red circular regions. Subsequently, the artifacts with an area of less than 400 pixels were deleted, obtaining solely the 12 ROIs in the binary image, which correspond to the nozzles.

The centroid of each ROI was computed using the 2D invariant moments of order $(p+q)$ for an image f of size $M \times N$, as follows

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^p f(x,y) \tag{4.1}$$

where $p, q \in \mathbb{Z}^+$. Then, the centroid coordinates of the nozzle NC_i are obtained as

$$\begin{aligned} \bar{x_i} &= \frac{m_{10}}{m_{00}} \\ \bar{y_i} &= \frac{m_{01}}{m_{00}} \end{aligned} \tag{4.2}$$

where $\bar{x_i}$ and $\bar{y_i}$ are the abscissa and ordinate, respectively.

Subsequently, the 12 centroids were grouped corresponding to the NLW nozzle rail and BLW nozzle array and then ordered from left to right. The grouping into nozzle classes was determined by the magnitude of their ordinates, whereas the sorting was based on the magnitude of the abscissas. In this stage, also the spray factor SF for each nozzle NC_i is computed. The SF is a distance equivalent to $\frac{1}{4}$ the diameter of the spraying area of the nozzles. Therefore, it is computing by dividing the distance $\frac{d}{4}$ from the centroid NC_i and NC_{i+1} in each nozzle array, as follows

$$SF = \frac{x_{i+1} - x_i}{4} \tag{4.3}$$

where x_i is the abscissa of the NC_i nozzle. The SF for the nozzle NC_6 in each nozzle rail was the same computed for NC_5.

In stage three of the process (Figure 4.22), the calibration image is divided into stripes. The stripes correspond to the spraying width of the nozzles, meaning that they will open for spraying the weeds that are inside the fringes. To compute each stripe S_i, first, the midpoint p_i and p_i', between the nozzle NC_i and nozzle NC_{i+1} in the NLW nozzle array and BLW nozzle array, respectively, is computed, as follows

$$p_i, p_i' = \frac{x_{i+1} - x_i}{2} \tag{4.4}$$

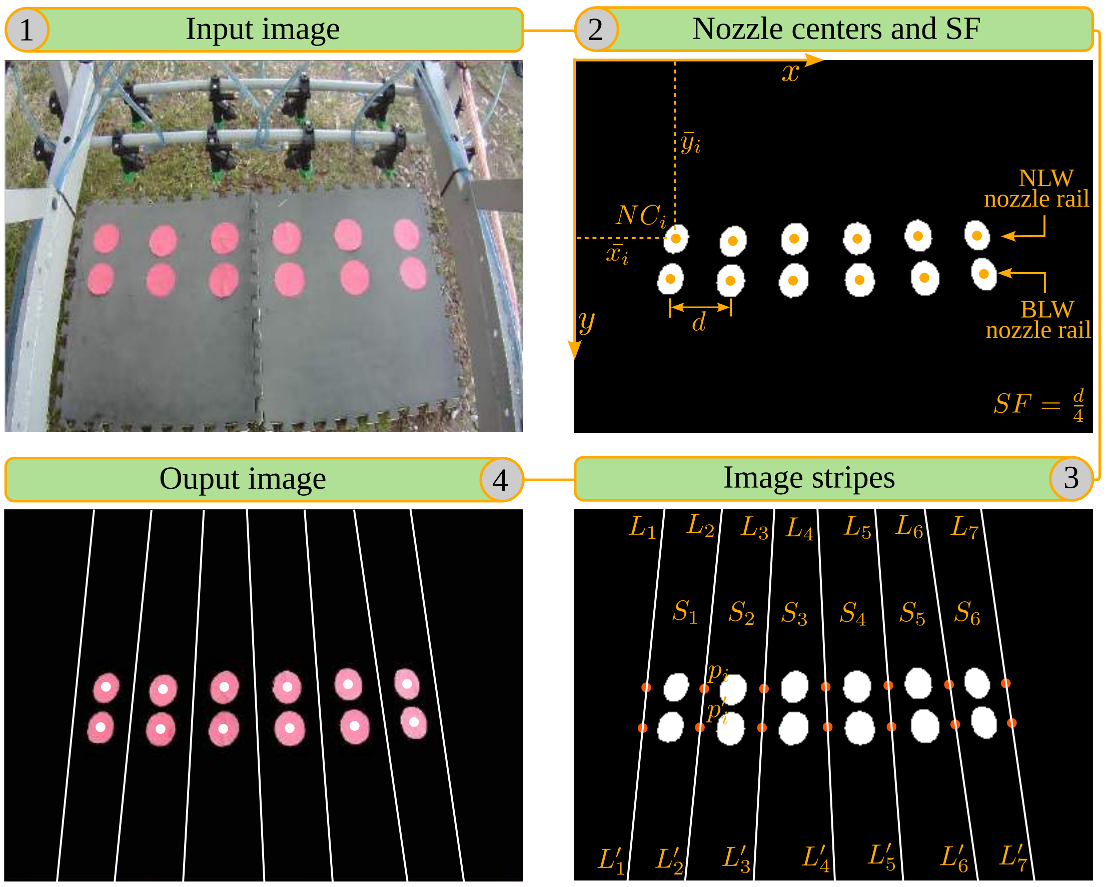


Figure 4.22: General illustration for obtaining the centers of the nozzles, the spray factor and spray stripes for the nozzles.

where x_i is the abscissa of the NC_i nozzle. Therefore, the coordinates of the points L_i and L'_i that represent the limits of each stripe are obtained by projecting the line that passes through p_i and p'_i using the point-slope equation for a line $(x_2 - x_1)m = (y_2 - y_1)$. However, the lines $L_1 - L'_1$ and $L_7 - L'_7$ are parallels of lines $L_2 - L'_2$ and $L_6 - L'_6$, respectively.

The fourth stage of the process involves visualizing the centers of each nozzle and the stripes in the SWS. The coding of this process is indicated in Algorithm 1 of Appendix A.

The computed nozzle centers, spray factors, and coordinates for striping the images are utilized in releasing the vision systems for weed detection in natural cornfields. Figure 4.23 illustrates the general procedure followed for vision release.

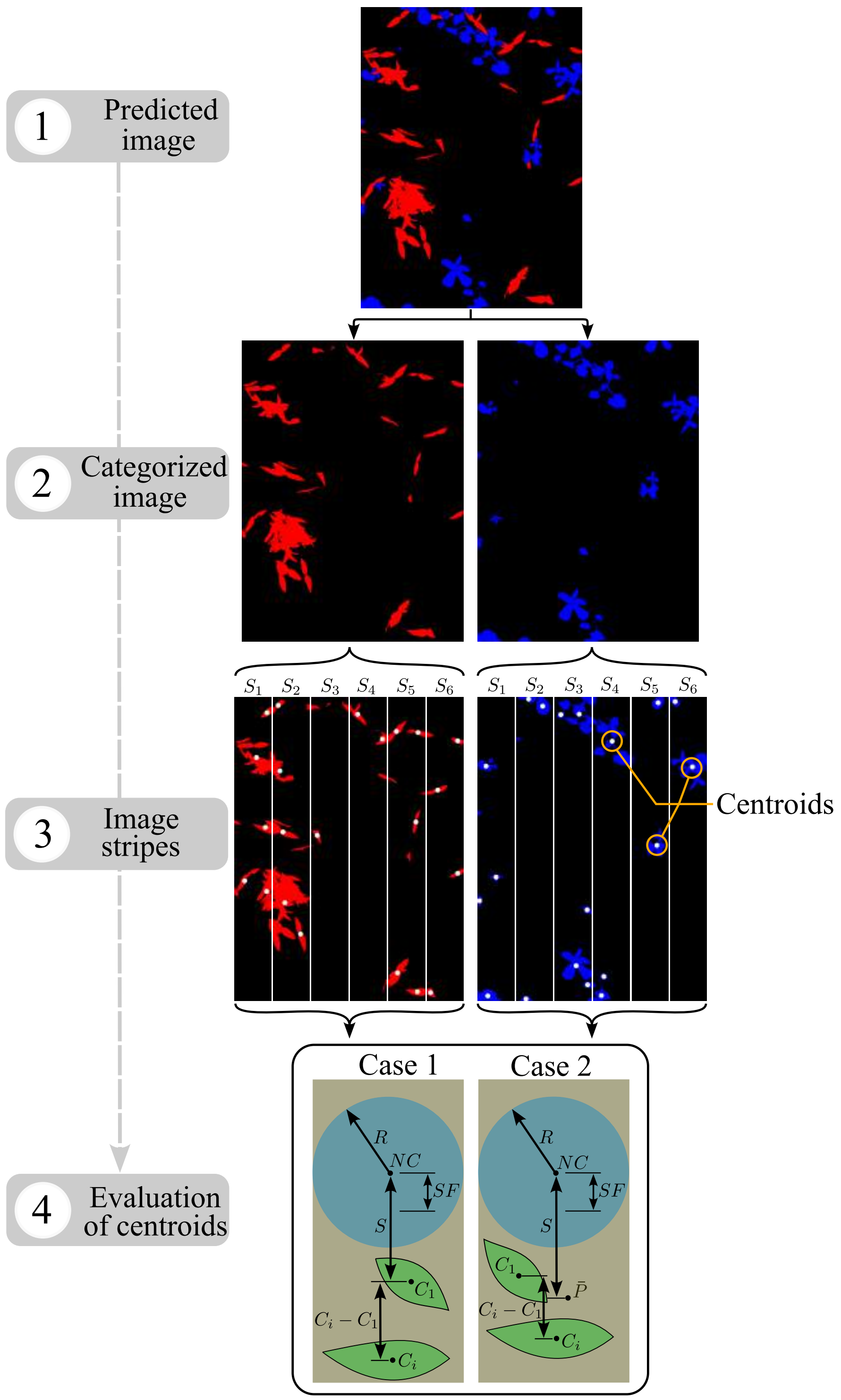


Figure 4.23: Illustration of the release process of the vision system for weed detection and control.

It is worth mentioning that, due to its outstanding performance, the SegFormer trained model was employed in this thesis for weed detection in natural cornfields. The image shown in stage one of the process represents the prediction mask generated by the model, where the red ROIs indicate plants of the class NLW and the blue ones indicate plants of the class BLW.

Subsequently, the image is classified into the NLW and BLW classes, resulting in a separate image for each class, as demonstrated in stage two. In stage three, the images are divided into fringes using the coordinates computed during the calibration stage. Additionally, the ordinate magnitude of the centroids for each ROI within every fringe is calculated.

Finally, in stage four, the ordinate magnitudes of the ROIs are evaluated in two cases to derive a binary vector $[0, 1]$ for activating the solenoid valves to open the nozzles at the precise time. The evaluation cases consider the ordinate of each nozzle center (NC) assigned to spray each fringe and their corresponding spray factor (SF). However, the ordinate values of the ROIs in each stripe are arranged according to their magnitude, and those with magnitudes lower than the nozzle's ordinate are removed to obtain a debugged list of ordinates. This is necessary because they lie outside the spraying zone of the nozzles and due to the forward direction the SWS follows in practice. Therefore, the cases are defined as shown below.

Case 1. The ordinate value C_1 of the closest ROI to the nozzle center (NC) in each stripe is compared with each ordinate C_i of the remaining ROIs in the same stripe. If for all i, the condition $C_i - C_1 > SF$ is true, then the distance S is defined as $S = C_1 - CN$.

Case 2. The ordinate value C_1 of the closest ROI to the nozzle center (NC) in each stripe is compared with each ordinate C_i of the remaining ROIs in the same stripe. If for at least one i, the condition $C_i - C_1 < SF$ is true, then the distance S is defined as $S = \bar{P} - CN$, where $\bar{P}$ is the average of the ordinates participating in the condition.

In this way, the nozzles will spray if the condition $SF \geq S \geq 0$ is true. The Algorithm 2 of Appendix A represents the code for the release of the vision system.

For the practical operation of the SWS in fields, a graphical user interface (GUI) was designed. This GUI consists of three main windows, as indicated in Figure 4.24. The main window, shown in Figure 4.24(a), is displayed upon the launching of the GUI. This main window directs the user to a calibration window and a spray window.

Concerning the calibration window (Figures 4.24(b) and (c)), the “Start” button initializes the webcam. Subsequently, the “Take picture” button allows users to capture an image of the calibration rug. After taking the picture, the “Calibrate” button becomes active. Pressing it activates the bottom sliding horizontal bars, which represent the upper and lower threshold values of the hue, saturation, and value channels. The values are used because the RGB image is transformed into the HSV color space. Internally, the image undergoes binarization and the small artifacts are removed, as explained previously. This process is done to isolate just the red circular regions, which are aligned with the nozzles. The real-time display of the updated masked RGB image is shown in the window, as illustrated in Figure 4.24(c). By pressing the “Save” button, the nozzle centroids, the spray factors of every nozzle, and the coordinates for splitting the predicted images are stored in a “txt” file. In case the user wants to re-calibrate the SWS, the “Reset” button initiates the process anew. Finally, the “Main menu” button directs the user back to the main window.

On the other hand, the spray window (Figure 4.24(a)) facilitates real-time visualization of predictions made by the model and enables observation of nozzle actions. In this way, the “Start” button initiates the release system explained in Algorithm 2, along with the electrical system. When the SWS reaches the border of the crop rows, the “Stop” button suspends both predictions and spraying. Subsequently, when the SWS is realigned with the next row objectives, pressing the ’Start’ button resumes the practice. Finally, the ’Main menu’ button directs the user back to the main window.

4.1.3. Field evaluation

This section presents the characterization of the experimental cornfield where the treatments, the SWS and the CWS, were evaluated. Also, the behavior of the

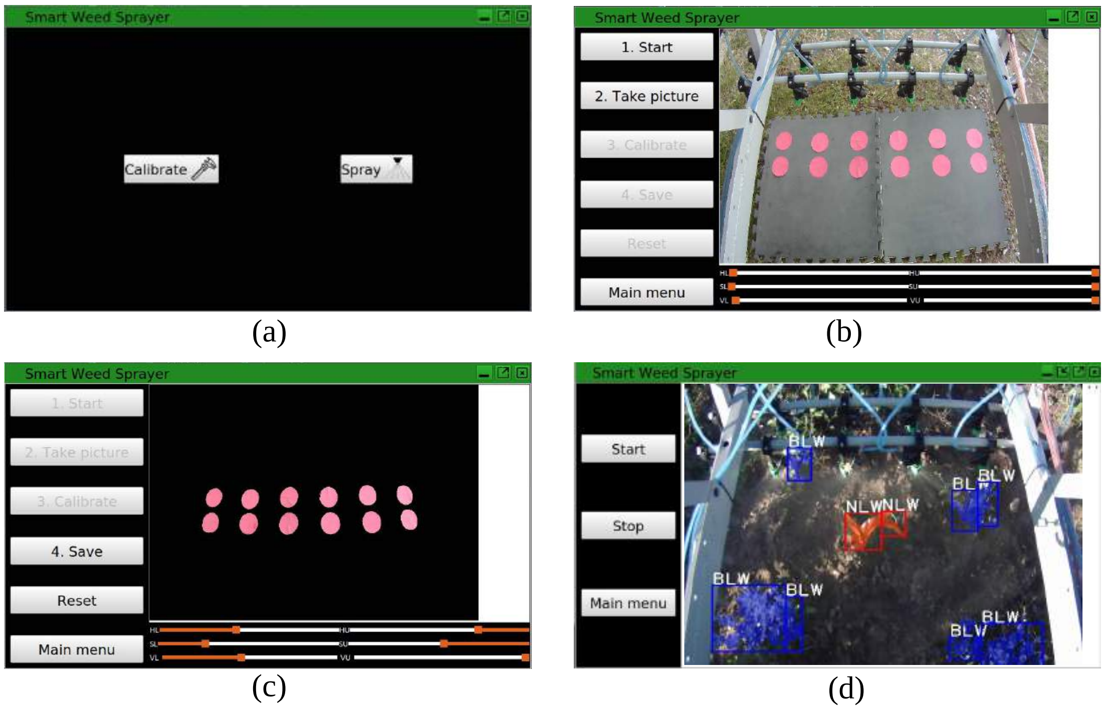


Figure 4.24: Graphical user interface for operation of the SWS.

response variables, herbicide expenditure per hectare ($L\,ha^{-1}$) and effectiveness in controlling weeds, attributed to each treatment are provided and compared.

Figure 4.25 is an image sample of the experimental cornfield. Concerning the characterization of this land area, the weed density was quantified as 59 $plt\,m^{-2}$. Additionally, it was found the 17 weed species listed in Table 4.8. Out of these, 13 species belong to the class BLW and four belong to the class NLW. Note that only two weed species of the BLW class from this list, namely *Amaranthus spinosus* and *Malva parviflora*, correspond to our original dataset on which the models were trained. However, all four NLW weed species from the training dataset were also found in the cornfield.

4.1.3.1. Herbicide expenditure per hectare

The spraying width of the SWS was measured to be 1.60m. Consequently, the area of each experimental unit for the SWS is approximately 240m^2, while the area of the experimental unit for the CWS is approximately 525m^2. Therefore,

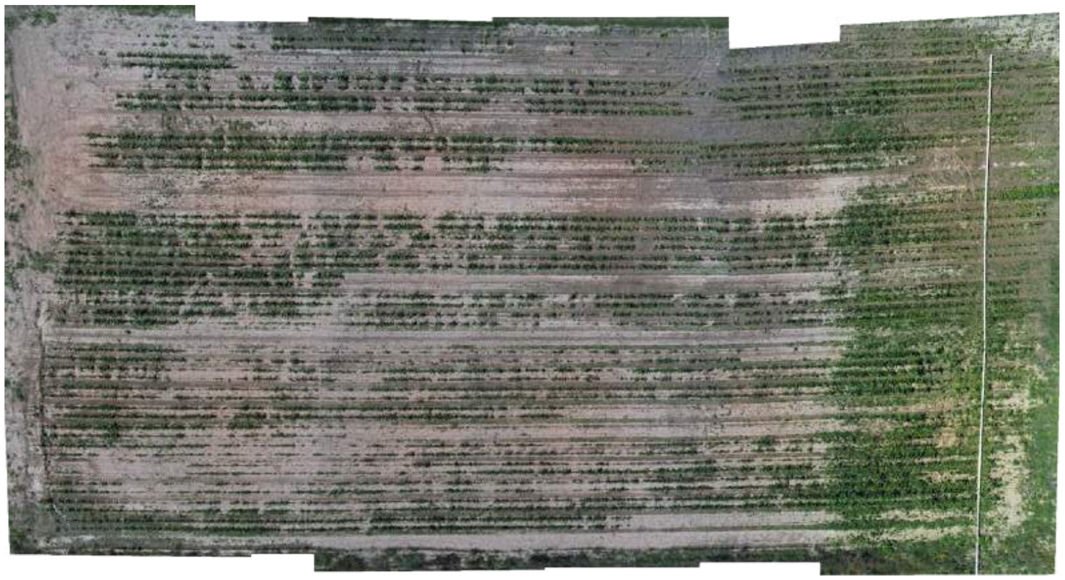


Figure 4.25: A sample orthomosaic captured from the experimental cornfield.

for the statistical analysis, the measured expenditure registered in the CWS was proportionally adjusted to match the area of the SWS's experimental unit for each replication.

As a result, significant differences ($P \leq 0.05$) were found among the treatments concerning the response variable "herbicide expenditure", which is the mixture of water and herbicide. The CWS exhibited a mean expenditure of $14.4712L$ whereas the SWS presented an expenditure of $7.8667L$.

Figure 4.26 shows the extrapolated herbicide wastage for each treatment per hectare. These counts indicate that the SWS has the potential to reduce the herbicide mixture usage by 45.64 %. This percentage reflects also both the savings in terms of active ingredient per hectare and the associated cost reduction related to herbicide purchasing through the use of the SWS.

4.1.3.2. Effectiveness in controlling weeds

A weed plant that has been sprayed with a systematic herbicide turns its color appearance from green to brown in a few days. This is because the sap flow

Table 4.8: Weed species in the experimental cornfield.

Class	Species
BLW	*Amaranthus spinosus*
	Argemone mexicana L.
	Bindes pilosa L.
	Chenopodium album L.
	Galinsoga parviflora Cav
	Glebionis coronaria L.
	Lysimachia arvensis L.
	Malva parviflora
	Sigesbeckia orientalis L.
	Sisymbrium irio L.
	Solanum elaeagnifolium
	Solanum rostratum Dunal
	Tribulus terrestris L.
NLW	*Cynodon dactylon*
	Cyperus esculentus
	Digitaria sanguinalis
	Eleusine indica

stops gradually, and also the sun rays evaporate the remaining water in the plant. Therefore, the color change helps to evaluate the effectiveness of herbicides.

In this thesis, as was mentioned earlier, the green color of weeds was used to evaluate the effectiveness of the treatments, CSW and SWS, in controlling the herbs. In this regard, Figure 4.27 shows the mean weed cover reduction observed during the days as the response to the treatments.

On November 19, the day the experiment was implemented, the weed cover represented 100 %. It is noteworthy that the weed cover decreased by 34.25 % in the area treated with CWS, while in the area treated with SWS, this variable was reduced by 44.08 %, five days after the spraying practice. The additional 10 % reduction in weed cover observed in the treated area with SWS at this stage is attributed to factors that were not blocks, since CWS performed a total cover of

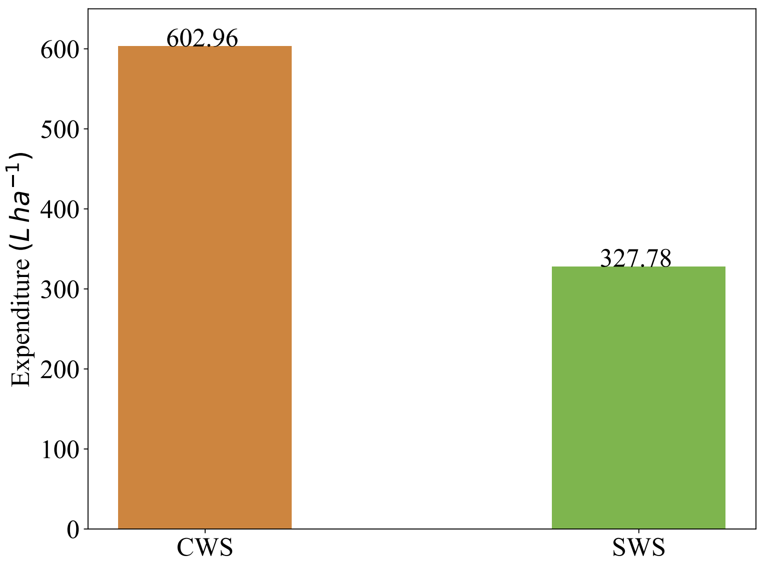


Figure 4.26: Extrapolated herbicide volume expenditure by the treatments per hectare.

the area. Finally, on November 29, ten days after the practice, both the CWS and the SWS exhibited a similar weed cover reduction within their respective treated areas, averaging 88.55 %. This data suggests that the effectiveness of both the CWS and SWS on controlling weeds is similar.

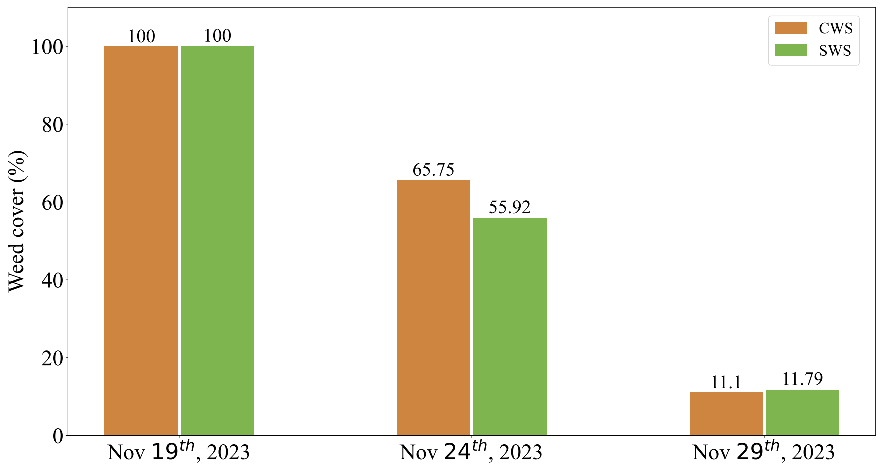


Figure 4.27: Weed cover reduction in the sprayed area by the treatments.

Finally, a sample of images showing the reduction of weed cover over the same area for each treatment during the day sequence is shown in Figure 4.28.

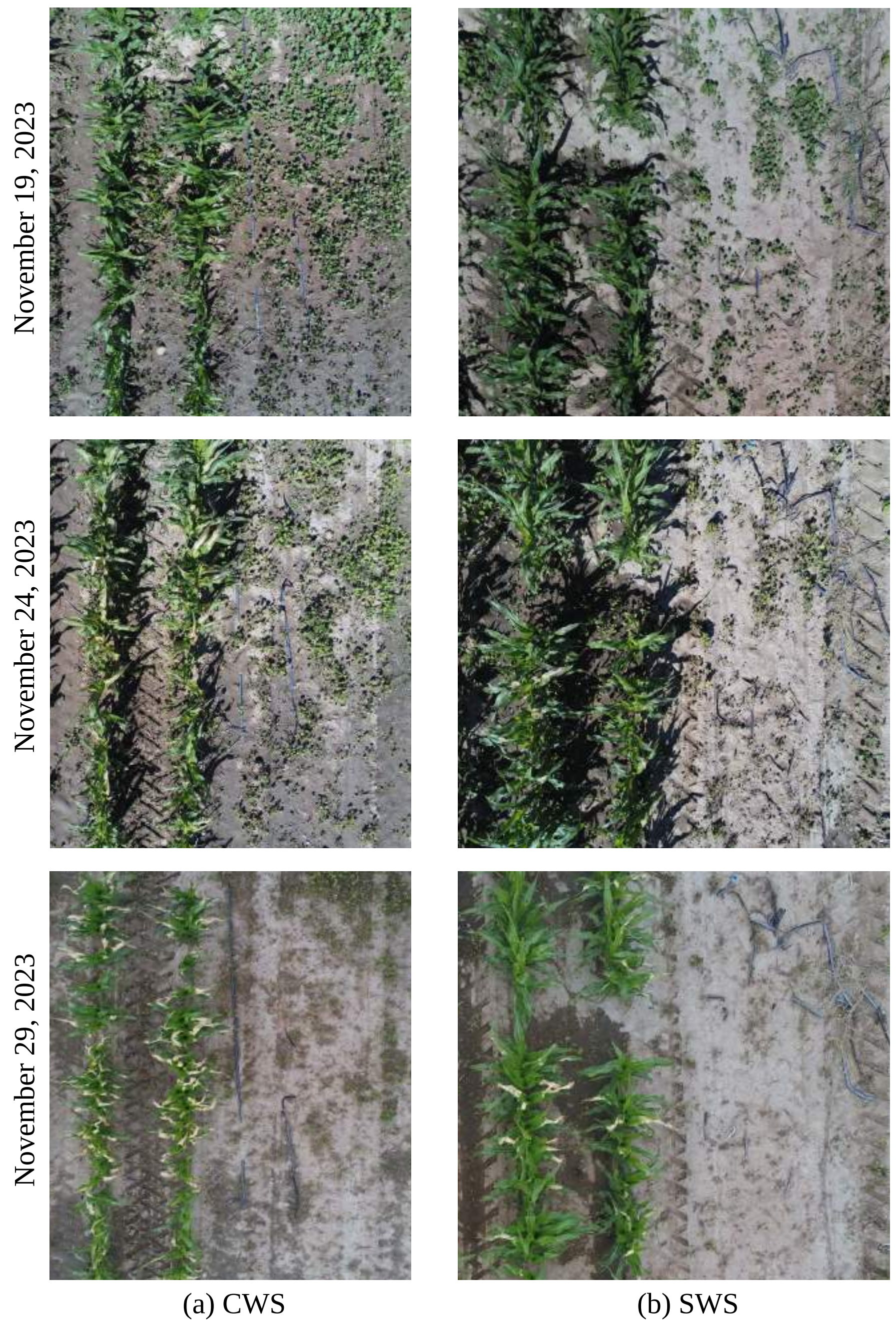


Figure 4.28: Image sample illustrating the reduction of weed cover for each treatment throughout the day sequence.

The first column corresponds to the sample images for the CWS and the second column for the SWS. On the other hand, the images in the first row from top to dawn depict the weed cover on the day the experiment was implemented. As observed, the weed plants are of green color. Following that, in the second row, the green tone of the herbs has undergone a slight change, indicating the action of the herbicide. In the third-row images, the weeds have practically transformed their appearance to brown, especially noticed in the area treated with the SWS. Nonetheless, it is observed that, due to the soil moisture, the area treated with CWS still exhibits some small regions with green color corresponding to weeds.

4.2. Discussion

This section presents a discussion of the findings related to the development of the vision system. It encompasses classification methods based on both shallow and DL, as well as segmentation techniques based on DL.

4.2.1. Crop/weed classification based on shallow and DL

Concerning the classification of the classes Crop, NLW and BLW of our dataset using $LBP_{P,R}^{riu2} + SVM$ we considered that our best SVM model gave an acceptable accuracy (83.04 %). In this scenario, the SVM model had to learn the complexity of the features associated with each distinct plant species and effectively relate them into a single class, rendering the classification task inherently complex.

Janahiraman et al. (2019) conducted an assessment of the models $LBP_{8,1}^{riu2}/SVM$ and $LBP_{16,2}^{riu2}/SVM$ for BLW classification using the Flavia dataset (Wu et al., 2007), yielding mean accuracies of 64.22 % and 75.49 %, respectively. When these same models were evaluated in the Swedich dataset (Söderkvist, 2001), another BLW dataset, the mean accuracy increased to 78.44 % and 85.56 % for $LBP_{8,1}^{riu2}/SVM$ and $LBP_{16,2}^{riu2}/SVM$, correspondingly. It is essential to note that both datasets were acquired under controlled light conditions, and the images exhibit uniform backgrounds. On the other hand, in the study presented by Chen et al. (2021), the authors documented a mean accuracy of 90.60 % for an SVM model trained with texture features of corn and weeds. This model was configured as $LBP_{8,1}^{riu2}/256 \times 256/64 \times 64$. Despite the dataset in Chen et al. (2021) being generated in actual field environments, consisting of 2000 images and encompassing the classes crop and weeds, there are some limitations. The weed class is further divided into two categories: NLW and BLW plant species. This composition reduces the model's ability to generalize effectively to unseen species of plants.

Conversely, the majority of literature works have predominantly focused on classifying individual plant species using CNNs. Under this scenarios, exceptional performances exceeding 97 % have been reported for VGG16 and VGG19 when classifying individual species. For instance, Le et al. (2020a) and Ahmad et al. (2021) classified four plant species, while Jiang et al. (2020) classified five species, achieving the mean performance with a relatively limited number of images for each species during model training. Contrastingly, the literature indicates that performance often degrades when training CNN models with a diverse set of plant species (Dyrmann et al., 2016; Olsen et al., 2019). Only some reported works in the literature on integrating weeds into classes NLW and BLW were found.

Yu et al. (2019b) noted that VGG16 achieved a mean accuracy of 99 % for classifying more than five broad-leaf weed species integrated into a single class over Dormant Bermuda grass. owever, this high accuracy was attributed to the uniform environment since the Dorman Bermuda grass appearance differed from the BLW appearance, triggering an easy weed differentiation. Conversely, in dos Santos Ferreira et al. (2019) study, VGG16 exhibited a mean accuracy of 83.4 % when trained with plants of soja, soil, and grass-broadleaf weeds, with the latter class including multiple plant species. This cause makes our work interesting since scarce information was found when CNN was trained with classes Crop, NLW, and BLW in real environments of cornfields.

As summary, utilizing a CNN-based approach has proven to outperform the classical ML approach consistently across all scenarios. Therefore, for the task of weed classification in early growth stages and natural environments, such as the one demonstrated in this study, achieving an average accuracy of 97.50 % implies that the CNN-based approach stands as the superior alternative for carrying out this task.

4.2.2. Semantic segmentation based on DL

In this thesis, convolutional neural networks (CNNs) and transformers were explored for semantic segmentation of the classes Crop, NLW, and BLW, aiming to implement further spatial localization of plants in cornfields. Both Mask R-CNN and Mask R-CNN-ASPP networks, specialized for localization and segmentation, were studied, along with U-Net-like networks. All of these models were equipped with ResNet50 and ResNet101 as the main CNN for feature extraction. Additionally, transformer architectures were investigated for segmenting our dataset.

The Mask R-CNN-ASPP models proposed in this work surpassed the performance of the original Mask R-CNN models. However, the segmentation task carried out for the two network configurations was not enough to adapt to our complex dataset. These networks initially identify potential ROIs in the image before proceeding with segmentation. What has been observed in this study is that the plant density and leaf occlusion present challenges in extracting ROIs belonging to the same plant species. In certain instances, foliage from other species is erroneously included in the same ROI, a phenomenon also noted by Liu et al. (2020). Additionally, plants that occupy a significant area in the image but have a low pixel count, such as the NLW class, often lead to segmentation failures in the model. Consequently, these factors are attributed to the lower average performance of the Mask R-CNN and Mask R-CNN-ASPP models.

The U-Net-like architecture were trained with solely RGB images and NIR images. It were found that the U-Net-like model performed better using solely RGB images. Indicating that this solution could be an economical alternative, as it only requires cameras that operate in the visible spectrum. In contrast, experiments involving NIR images showed the lowest performance, suggesting that NIR channels do not provide sufficiently distinguishable features for the model to correlate them effectively with RGB images. However, when the architecture was trained with a combination of NIR and RGB images, a slight performance improvement was observed. This enhancement is attributed to the sharing features among the portion of the training dataset with the testing portion, which exclusively includes RGB images. It is noteworthy that in the literature, the effects of multispectral channels on image segmentation have been studied in works such as Fawakherji et al. (2021); Lottes et al. (2020, 2018, 2016, 2017); Sahin et al. (2023). However, testing images in these studies also contain multispectral channels. The adoption of multispectral cameras, though explored in the literature, is not considered cost-effective at the time of developing this thesis due to the still high prices of such cameras.

Contrasting transformers and CNNs for segmentation using solely RGB images, we observed in this work that the transformer SegFormer performed better in segmenting our dataset. Subsequently, the U-Net-like-ResNet101 model ranked second when the images were divided into patches. Segmenting the patches extracted from the input images, without altering the original size, may aid in preserving essential class features. Finally, the U-Net-like-ResNet101 model, when trained exclusively with resized RGB images, achieved acceptable performance, as shown in Figure 4.29. In summary, SegFormer obtained a DSC and mIoU of 5.97 % and 8.7 %, respectively, higher than those achieved by the U-Net-like model when the images were resized. Comparing SegFormer with the U-Net-like model when images were divided into patches, it was 2.76 % and 4.74 % superior in DSC and mIoU, respectively. Notably, these metrics' magnitudes are acceptable, surpassing the performance of other works reported in the literature that focused on segmenting corn and weed plants in natural environments. For instance, Quan et al. (2021) achieved a mIoU of 64.2 % when segmenting *Solanum nigrum*, *Echinochloa crus-galli*, and *Abutilon theophrasti*. In the work of Picon et al. (2022), a DSC of 25.32 % has been reported when segmenting six weed species and corn plants.

Detecting common weeds in corn fields poses challenges, therefore, few studies have addressed this issue in natural conditions with high-density plants using semantic segmentation approaches. In the research by Fawakherji et al. (2020), they evaluated the original U-Net architecture (Ronneberger et al., 2015) and U-Net with VGG16 network (Simonyan and Zisserman, 2015) as a backbone (U-Net-VGG16). The reported mIoU for U-Net and U-Net-VGG16 was 62 % and 64 %, respectively, when trained with a *Sunflower* dataset and tested on combined datasets *Carrots* and *SugarBeets*. The classes included soil, crop, and weed.

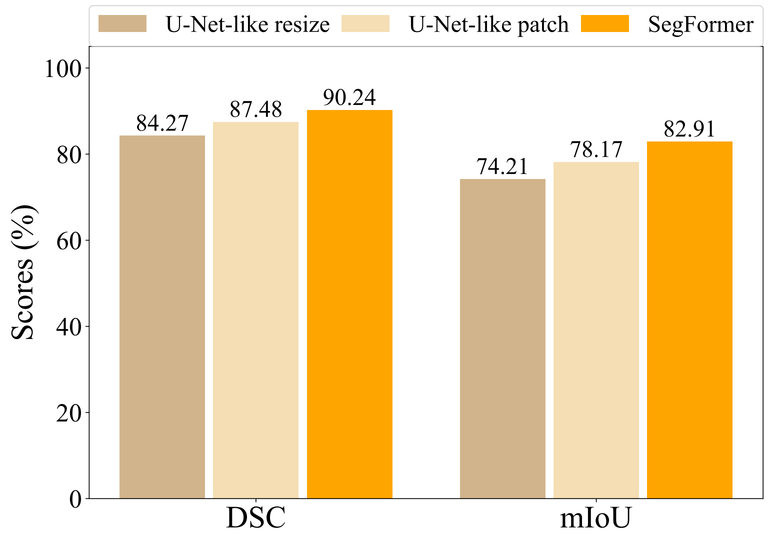


Figure 4.29: Average performance of the better three models for corn plants and weeds segmentation under natural environments.

They further evaluated U-Net-VGG16 on individual datasets *SugarBeets*, *Stuttgart*, *Carrots*, and *Sunflower*, reporting mIoU values of 71 %, 45 %, 35 %, and 39 %, respectively. Although their study did not focus on corn crops, the databases were generated under natural conditions. In comparison, the mIoU of our best model, SegFormer, is 18.91 % higher than the U-Net-VGG16 model reported by Fawakherji et al. (2020).

Furthermore, it's worth noting that our trained model exhibits potential for segmenting other monocotyledon and dicotyledon plant species. The classes NLW and BLW, for which the architecture was trained, encompass four species from each group, each at distinct growth stages, and also a group of unknown species into these classes. Additionally, the field variability was sufficiently diverse, enhancing the model's adaptability.

Other related works on semantic segmentation of crop plants and weeds are presented in Table 4.9. Even though the crops and trained architectures differ from ours, they also share the complexity of training the deep learning models using datasets acquired in natural environments. Therefore, the parameters dataset size, number of plant species in the dataset, DSC, and the mIoU have been highlighted to contrast them with our work. In this case, our work stands out

Table 4.9: Performance of related works upon semantic segmentation of crop/weed in natural environments.

Plant species	DS	**Model**	**mIoU**	**DSC**	**Reference**
Rice seedling *Sagittaria trifolia*	28	SegNet-VGG16 FCN UNet	91.80 % 53.80 % 53.00 %	– – –	Ma et al. (2019)
Rice seedling *Sagittaria trifolia*	224	UNet SegNet FCN-8s DeepLabV3 CED-Net	59.67 % 67.41 % 54.78 % 67.60 % 71.05 %	74.74 % 80.53 % 70.78 % 80.67 % 83.08 %	Khan et al. (2020)
Rice Broadleaf weeds and narrowleaf weeds. Species are not specified	1690	PSPNet-ResNet50 UNet-ResNet50 SegNet-VGG16	62.44 % 51.35 % 31.88 %	– – –	Kamath et al. (2022)
Wheat crop Not included	190	DeepLabV3+ - ResNet50	77.50 %	86.30 %	Zenkl et al. (2022)
Corn, *Setaria verticillata*, *Digitaria sanguinalis*, *Echinochloa crus-galli*, *Abutilon theophrasti*, *Chenopodium albums* and *Amaranthus retroflexus*	1679	PSPNet-ResNet50 Dual PSPNet-ReSNet50	– –	45.33 % 47.97 %	Picon et al. (2022)
Grass *Trifolium repens*, *Ambrosia artemisiifolia*, *Digitaria*, *Taraxacum*, *Glechoma hederacea*, *Chenopodium album*, *Amaranthus*, *Plantago asiatica L.*, *Festuca arundinacea* and *Unknown weeds*	1006	Swin Transformer SegFormer Segmenter	65.41 % 65.74 % 59.24 %	– – –	Jiang et al. (2023)

DS - dataset size in number of images; mIoU - mean intersection over union; DSC - dice similarity coefficient.

from the others because a large dataset has been used, which contains nine plant species and 12,887 images. Additionally, the metrics DSC and mIoU are also better compared with the rest of the works presented in this table.

Chapter 5

Conclusions and Recommendations

5.1. Conclusions

In this thesis, the development of a vision system powered by deep learning techniques to detect weeds in natural cornfields is proposed. This system is later implemented on a mechatronic platform to enable real-time weed control via the application of herbicides. The system's effectiveness is evaluated through an experiment that measures herbicide expenditure and its efficacy in combating weeds.

To train deep learning models, a dataset of 12,887 images was created. The images consist of corn plants (Crop), four NLW species (NLW), and four BLW species, all of which were captured in natural cornfield environments at various growth stages. The plant species depicted in the images were manually annotated at the pixel level. The dataset includes 10,575 images captured using a visual spectrum camera and 2,312 images taken with a multispectral camera with NIR and Red Edge channels. The dataset comprises 28,507, 41,795, and 52,541 plant instances in the Crop, NLW, and BLW classes, respectively. The multispectral channels contain 10,039 instances of the Crop class and 23,160 and 29,920 instances of various species in the NLW and BLW classes, respectively.

Throughout the development of our vision, we have explored four distinct approaches. Initially, we investigated the potential of classification options by utilizing classical descriptors and shallow classifiers while also studying DL classifiers. Subsequently, we addressed the issue by exclusively utilizing segmentation CNNs trained with both RGB and multispectral images. Moreover, we have examined the possible synergy between segmentation CNNs and classification CNNs. Lastly, we have employed an alternative vision strategy utilizing transformer architectures to tackle the problem at hand.

Experimental results indicate that the better approach for weed classification using shallow learning (LBP+SVM) achieved an accuracy rate of 83.04 %. However, the best deep learning approach, which utilized the VGG16 network, yielded a significantly higher accuracy rate of 97.93 %, suggesting that deep learning is more effective than shallow learning. It is worth noting, however, that relying solely on classification is insufficient for weed control, as it fails to provide spatial information about the location of the weeds.

An approach based on semantic segmentation has been implemented to address the spatial localization of plants. The evaluation study of the performance of the U-Net-like-ResNet101 model using NIR images was conducted. First, the model was trained with NIR images and evaluated on RGB images. Next, the model was trained with NIR images, and then transfer learning was used to retrain it with RGB images. Finally, the model was trained with a combination of NIR and RGB images. However, experimental results show that these approaches performed below the performance achieved by solely training the model with RGB images.

Multiple models have been tested using exclusively RGB images, and the U-Net-based network has shown superior performance over the others. Specifically, the U-Net-like-ResNet101 model achieved a DSC of 84.27 % and a mIoU of 74.21 %. Nonetheless, the transformer SegFormer surpassed U-Net-like-ResNet101 evaluated under the same conditions, which obtained 90.24 % of DSC and 82.91 % of mIoU.

The SWS mechatronic platform has been designed and built to incorporate the crop/weed identification system. The SWS includes two herbicide tanks, one formulated explicitly for controlling narrow-leaf weeds and the other for broadleaf weeds control. The SWS also features a sprinkler rail with six nozzles for each herbicide tank, each controlled by solenoid valves, which can open or close depending on the need. Thus, if the vision system detects weeds within the spatial regions corresponding to the nozzles, the solenoid valves open, allowing herbicide flow for precise spraying of the targets.

The SWS was tested in authentic cornfield conditions and compared to a CWS. The results demonstrated a statistically significant difference with a $p-value \leq 0.05$ in relation to herbicide expenditure per hectare (L/ha). These findings suggest that the SWS has the potential to reduce herbicide mixture usage by 45.64 %. In terms of effectiveness in weed control, the area treated with the CWS experienced a reduction of 34.25 % based on weed cover observed on the day of the practice, while the SWS-treated area exhibited a higher reduction of 44.08 % five days post-spraying.

In future work, we plan to expand the dataset by incorporating more plant species per class, all captured in authentic cornfields. Additionally, we aim to assess the effectiveness of the SWS by introducing controlled weed densities with more replication per treatment across diverse cornfield locations within the state of Aguascalientes.

5.2. Recommendations

Deep learning architectures typically require extensive datasets to train models capable of generalizing to unseen data. Therefore, our recommendations go in this direction.

It is advisable to augment the crop and weeds dataset by increasing the number of instances per plant species and the number of plant species per class, including Crop, NLW, and BLW. Furthermore, capturing images in real cornfields is essential to introduce the greatest possible variability to the dataset.

Also, to reduce the annotation time, it is better to grow control weed areas exclusively with with NLW or BLW multi-plant species. Then, during the annotation stage, all green regions in the image will correspond solely to one of these big classes.

Images captured in a single shot exhibit superior quality and do not suffer from the blurriness often associated with frames extracted from videos. However, in the deployment of a vision system, video images are commonly encountered.

Consequently, it is advisable to record videos in cornfields and subsequently extract images from them for the annotation of plant species. This approach allows the model to learn essential features and improves the likelihood of accurate plant recognition during the deployment stage.

References

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer Nature. 43, 48, 49, 52

Ahmad, A., Saraswat, D., Aggarwal, V., Etienne, A., and Hancock, B. (2021). Performance of deep learning models for classifying and detecting common weeds in corn and soybean production systems. *Comput. Electron. Agric.*, 184:106081. 23, 25, 77, 161

Akbarzadeh, S., Paap, A., Ahderom, S., Apopei, B., and Alameh, K. (2018). Plant discrimination by support vector machine classifier based on spectral reflectance. *Comput. Electron. Agric.*, 148:250–258. 6

Alcántara-de la Cruz, R., Cruz-Hipolito, H. E., Domínguez-Valenzuela, J. A., and De Prado, R. (2021). Glyphosate ban in mexico: potential impacts on agriculture and weed management. *Pest Manag Sci*, 77:3820–3831. 3

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. (2021). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *J Big Data.*, 8(1):53. 29, 78

Amjoud, A. B. and Amrouch, M. (2023). Object detection using deep learning, cnns and vision transformers: A review. *IEEE Access*, 11:35479–35516. 22

ASABE (2006). *ASABE Standards.* ASAE S217.12 DEC01 Three-Point Free-Link Attachment for Hitching Implements to Agricultural Wheel Tractors. St. Joseph, MI: ASABE. 144

Asad, M. H. and Bais, A. (2020). Weed detection in canola fields using maximum likelihood classification and deep convolutional neural network. *Information Processing in Agriculture*, 7:535–545. 26

Asif, M., Amir, S., Israr, A., and Faraz, M. (2010). A vision system for autonomous weed detection robot. *International Journal of Computer and Electrical Engineering*, 2(3):1793–8163. 13

Bakhshipour, A. and Jafari, A. (2018). Evaluation of support vector machine and artificial neural networks in weed detection using shape features. *Computer and Electronics in Agriculture*, 145:153–160. 9, 11, 14, 15

Bakhshipour, A., Jafari, A., Nassiri, S. M., and Zare, D. (2017). Weed segmentation using texture features extracted from wavelet sub-images. *Biosystem Engineering*, 157:1–12. 12, 15

Barrero, O. and Perdomo, S. A. (2018). Rgb and multispectral uav image fusion for gramineae weed detection in rice field. *Precis. Agric.*, 19:809–822. 9

Biller, R. H. (1998). Reduced input of herbicides by use of optoelectronic sensors. *Journal of Agricultural Engineering Research*, 71:357–362. 7

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer Nature. 31, 45, 47

Burgos-Artizzu, X. P., Ribeiro, A., Guijarro, M., and Pajares, G. (2011). Real-time image processing for crop/weed discrimination in maize fields. *Computer and Electronics in Agriculture*, 75(2):337–346. 9

Cao, H., Wang, Y., Chen, J., Jiang, D., Zhang, X., Tian, Q., and Wang, M. (2021). Swin-unet: Unet-like pure transformer for medical image segmentation. *arXiv*, page arXiv:2105.05537. 99

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. *arXiv*, page arXiv:2005.12872. 100

Chaki, J., Parekh, R., and Bhattacharya, S. (2015). Plant leaf recognition using texture and shape features with neural classifiers. *Pattern Recognition Letters*, 58:61–68. 12

Champ, J., Mora-Fallas, A., Goëau, H., Mata-Montero, E., Bonnet, P., and Joly, A. (2020). Instance segmentation for the fine detection of crop and weed plants by precision agricultural robots. *Applications in Plant Sciences*, 8(7):e11373. 19

Chebrolu, N., Lottes, P., Schaefer, A., Winterhalter, W., Burgard, W., and Stachniss, C. (2017). Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. *Int. J. Robot. Res.*, 36(10):1045–1052. 19, 20, 21

Chen, J., Wang, H., Zhang, H., Luo, T., Wei, D., Long, T., and Wang, Z. (2022). Weed detection in sesame fields using a yolo model with an enhanced attention mechanism and feature fusion. *Computer and Electronics in Agriculture*, 202:107412. 25

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 834–848. 84

Chen, Y., Wu, Z., Zhao, B., Fan, C., and Shi, S. (2021). Weed and corn seedling detection in field based on multi feature fusion and support vector machine. *Sensors*, 21:212. 8, 9, 14, 15, 160

Chen, Y., Zhao, B., Li, S., Liu, L., Yuan, Y., and Zhang, Y. (2015). Weed reverse positioning method and experiment based on multi-feature. *Trans. Chin. Soc. Agric. Mach.*, 46:257–262. 11

Cheng, H., Jiang, X., Sun, Y., and Wang, J. (2001). Color image segmentation: advances and prospects. *Pattern Recognition*, 34(12):2259–2281. 70

Che'Ya, N. N. (2016). *Site-Specific Weed Management Using Remote Sensing.* PhD thesis, The University of Queensland. 5

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 74, 77, 78

Christensen, S., Heisel, T., Walter, A. M., and Graglia, E. (2003). A decision algorithm for patch spraying. *Weed Res.*, 43:276–284. 3

Dadashzadeh, M., Abbaspour-Gilandeh, Y., Mesri-Gundoshmian, T., Sabzi, S., Hernández-Hernández, J. L., Hernández-Hernández, M., and Arribas, J. I. (2020). Weed classification for site-specific weed management using an automated stereo computer-vision machine-learning system in rice fields. *Plants*, 9:559. 14

Das, M. and Bais, A. (2021). Deepveg: Deep learning model for segmentation of weed, canola, and canola flea beetle damage. *IEEE Access*, 9:119367–119380. 26

De Rainville, F. M., Durand, A., Fortin, F. A., Tanguy, K., Maldague, X., Panneton, B., and Simard, M. J. (2014). Bayesian classification and unsupervised learning for isolating weeds in row crops. *Pattern Anal Applic*, 17:401–414. 14

de Souza, M. F., do Amaral, L. R., de Medeiros Oliveira, S. R., Coutinho, M. A. N., and Netto, C. F. (2020). Spectral differentiation of sugarcane from weeds. *Biosyst. Eng.*, 190:41–46. 6

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Miami, FL, USA. 16, 90, 96

Deng, W., Huang, Y., Zhao, C., Chen, L., and Wang, X. (2016). Bayesian discriminant analysis of plant leaf hyperspectral reflectance for identification of weeds from cabbages. *Afr. J. Agric. Res.*, 11:551–562. 6

Deng, W., Huang, Y., Zhao, C., and Wang, X. (2014). Discrimination of crop and weeds on visible and visible/near-infrared spectrums using support vector machine, artificial neural network and decision tree. *Sens. Transducers*, 26:26–34. 6, 14

dos Santos Ferreira, A., Freitas, D. M., da Silva, G. G., Pistori, H., and Folhes, M. T. (2019). Unsupervised deep learning and semi-automatic data labeling in weed discrimination. *Computers and Electronics in Agriculture*, 165:104963. 161

dos Santos Ferreira, A., Matte Freitas, D., Gonçalves da Silva, G., Pistori, H., and Theophilo Folhes, M. (2017). Weed detection in soybean crops using convnets. *Comput. Electron. Agric.*, 143:314—-324. 14, 15, 19, 20, 23

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16 x 16 words: Transfoers for image recognition at scale. *arXiv*, page arXiv:2010.11929. 57, 58, 100

Dutta, A. and Zisserman, A. (2019). The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, New York, NY, USA. ACM. 67, 69, 109

Dyrmann, M., Jørgensen, R. N., and Midtiby, H. S. (2017). Roboweedsupport–detection of weed locations in leaf occluded cereal crops using a fully convolutional neural network. *Advances in AnimalBiosciences: Precision Agriculture (ECPA)*, 8:842–847. 25

Dyrmann, M., Karstoft, H., and Midtiby, H. S. (2016). Plant species classification using deep convolutional neural network. *Biosystems Engineering*, 151:72–80. 20, 161

Espejo-Garcia, B., Mylonas, N., Athanasakos, L., Fountas, S., and Vasilakoglou, I. (2020). Towards weeds identification assistance through transfer learning. *Computers and Electronics in Agriculture*, 171:105306. 9, 18, 19, 20, 77, 114

Espejo-Garcia, B., Mylonas, N., Athanasakos, L., Vali, E., and Fountas, S. (2021). Combining generative adversarial networks and agricultural transfer learning for weeds identification. *Biosystems Engineering*, 204:79–89. 18

FAO (2023). Crops and livestock products. `https://www.fao.org/faostat/en/#data/QCL` [Accessed: 25 July 2023]. 1

Fawakherji, M., Potena, C., Pretto, A., Bloisi, D. D., and Nardi, D. (2021). Multi-spectral image synthesis for crop/weed segmentation in precision farming. *Robotics and Autonomous Systems*, 146:103861. 19, 20, 163

Fawakherji, M., Youssef, A., Bloisi, D. D., Pretto, A., and Nardi, D. (2020). Crop and weed classification using pixel-wise segmentation on ground and aerial images. *Int. J. of Robotics Computing*, 2:39–57. 20, 163, 164

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. 108

Gao, J., French, A. P., Pound, M. P., He, Y., Pridmore, T. P., and Pieters, J. G. (2020). Deep convolutional neural networks for image-based *Convolvulus sepium* detection in sugar beet fields. *Plant Methods*, 16:29. 19, 25

Gao, J., Nuyttens, D., Lootens, P., He, Y., and Pieters, J. G. (2018). Recognising weeds in a maize crop using a random forest machine-learning algorithm and near-infrared snapshot mosaic hyperspectral imagery. *Biosystems Engineering*, 170:39–50. 2, 10, 14

Garibaldi-Márquez, F., Flores, G., Mercado-Ravell, D. A., Ramírez-Pedraza, A., and Valentín-Coronado, L. M. (2022). Weed classification from natural corn field-multi-plant images based on shallow and deep learning. *Sensors*, 22:3021. 2, 8

Genze, N., Ajekwe, R., Güreli, Z., Haselbeck, F., Grieb, M., and Grimm, D. G. (2022). Deep learning-based early weed segmentation using motion blurred uav images of sorghum fields. *Computer and Electronics in Agriculture*, 202:107388. 17

Giselsson, T. M., Jørgensen, R. N., Jensen, P. K., Dyrmann, M., and Midtiby, H. S. (2017). A public image database for benchmark of plant seedling classification algorithms. *arXiv*, page arXiv:1711.05458. 19

González, C. R. and Woods, E. R. (2018). *Digital Image Processing.* Pearson: New York, NY, USA. 72

Guerrero, J. M., Pajares, G., Montalvo, M., Romeo, J., and Guijarro, M. (2012). Support vector machines for crop/weeds identification in maize fields. *Expert Syst. Appl.*, 39:11149–11155. 9

Hamouchene, I., Aouat, S., and Lacheheb, H. (2014). Texture segmentation and matching using lbp and glcm matrix. In *Intelligent Systems for Science and Information: Extended and Selected Results from the Science and Information Conference 2013*, pages 389–407. Cham, Switzerland. 12, 32, 33

Hamuda, E., Glavin, M., and Jones, E. (2016). A survey of image processing techniques for plant extraction and segmentation in the field. *Computer and Electronics in Agriculture*, 125:184–199. 2, 8

Haralick, R. M., Shanmugam, K., and Dinstein, I. (1973). Textural features for image classification. In *IEEE Transactions on Systems, Man, and Cybernetics*, pages 610–621. 12, 74, 93

Haralick, R. M. and Shapiro, L. G. (1992). *Computer and Robot Vision.* Addison-Wesley Publishing Company, Inc.: Boston, MA, USA. 73, 74, 93

Haug, S. and Ostermann, J. (2015). A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks. In *Computer Vision - ECCV 2014. Lecture Notes in Computer Science (LNCS).* 19

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask r-cnn. In *IEEE In International Conference on Computer Vision (ICCV)*, pages 2980–2988. December, Venice, Italy. 83, 91

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 85, 90

Hecht-Nielsen (1989). Theory of the backpropagation neural network. In *International 1989 Joint Conference on Neural Networks*, pages 593–605. Washington, DC, USA. 42

Herrera, P. J., Dorado, J., and Ángel Ribeiro (2014). A novel approach for weed type classification based on shape descriptors and a fuzzy decision-making method. *Sensors*, 14:15304–15324. 11, 15

Herrmann, I., Shapira, U., Kinast, S., Karnieli, A., and Bonfi, D. J. (2013). Ground-level hyperspectral imagery for detecting weeds in wheat field. *Prescision Agric.*, 14:637—-659. 10, 15

Hu, M. K. (1962). Visual pattern recognition by moment invariants. In *IRE Transactions on Information Theory*, pages 179–187. 11

Hu, X., Chu, L., and Pei, J. (2021). Model complexity of deep learning: a survey. *Knowl Inf Syst*, 2021:2585—-2619. 16

Hunt, E. R., Cavigelli, M., Daughtry, C. S. T., and c L Walthall (2005). Evaluation of digital photography from model aircraft for remote sensing of crop biomass and nitrogen status. *Precis. Agric.*, 6:359–378. 9

Imoloame, E. O. (2017). Evaluation of herbicide mixtures and manual weed control method in maize (zea mays l.) production in the southern guinea agro-ecology of nigeria. *Cogent Food & Agriculture*, 3:1375378. 3

Janahiraman, T. V., Yee, L. K., Der, C. S., and Aris, H. (2019). Leaf classification using local binary pattern and histogram of oriented gradients. In *7th International Conference on Smart Computing & Communications (ICSCC)*. 160

Janneh, L. L., Zhang, Y., Cui, Z., and Yang, Y. (2023). Multi-level feature re-weighted fusion for the semantic segmentation of crops and weeds. *Journal of King Saud University - Computer and Information Sciences*, 35:101545. 3, 20

Jeon, H. Y., Tian, L. F., and Zhu, H. (2011). Robust crop and weed segmentation under un-controlled outdoor illumination. *Sensors*, 11:6270–6283. 9

Jiang, H., Zhang, C., Qiao, Y., Zhang, Z., Zhang, W., and Song, C. (2020). Cnn feature based graph convolutional network for weed and crop recognition in smart farming. *Computer and Electronics in Agriculture*, 174:105450. 19, 20, 23, 161

Jiang, K., Afzaal, U., and Lee, J. (2023). Transformer-based weed segmentation for grass management. *Sensors*, 23:65. 18, 21, 27, 99, 165

JIN, X., CHE, J., and CHEN, Y. (2021). Weed identification using deep learning and image processing in vegetable plantation. *IEEE Access*, 9:10940–10950. 9

Kamath, R., Balachandra, M., and Prabhu, S. (2020). Crop and weed discrimination using laws' texture masks. *Int J Agric & Biol Eng*, 13(1):191–197. 14

Kamath, R., Balachandra, M., Vardhan, A., and Maheshwari, U. (2022). Classification of paddy crop and weeds using semantic segmentation. *Cogent Engineering*, 9:2018791. 3, 26, 165

Kamilaris, A. and Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computer and Electronics in Agriculture*, 147:70–90. 16

Karadöl, H., Aybek, A., and Ücgül, M. (2020). Development of an automatic system to detect and spray herbicides in corn fields. *Journal of Agricultural Sciences*, 26:190–200. 9

Karimi, Y., Prasher, S., Patel, R., and Kim, S. (2006). Application of support vector machine technology for weed and nitrogen stress detection in corn. *Comput Electron Agric.*, 51:99–109. 5

KC, K., Yin, Z., Li, D., and Wu, Z. (2021). Impacts of background removal on convolutional neural networks for plant disease classification in-situ. *Agriculture*, 11(9):827. 20

Khan, A., Ilyas, T., Umraiz, M., Mannan, Z. I., and Kim, H. (2020). Ced-net: Crops and weeds segmentation for smart farming using a small cascaded encoder-decoder architecture. *Electronics*, 9:1602. 26, 165

Khan, S., Rahmani, H., Shah, S. A. A., and Bennamoun, M. (2018). *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan and Claypool Publishers. 48, 52

Knežević, D., Božić, S., Božić, D., and Vrbničanin, S. (2021). Critical time for weed removal in corn as influenced by planting pattern and pre herbicides. *Agriculture*, 11:587. 2

Konduri, V. S., Vandal, T. J., Ganguly, S., and Ganguly, A. R. (2020). Data science for weather impacts on crop yield. *Front. Sustain. Food Syst.*, 4:52. 1

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *25th International Conference on Neural Information Processing Systems*, pages 1097–1105. (Lake Tahoe, NV. 16, 54

Kyllo, K. P. (2003). *NASA funded research on agricultural remote sensing.* Department of Space Studies, University of North Dakota. 4, 5

Lam, S. W. C. (1996). Texture feature extraction using gray level gradient based co-occurence matrices. In *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems*, pages 267–271. 12

Lameski, P., Zdravevski, E., Trajkovik, V., and Kulakov, A. (2017). Weed detection dataset with rgb images taken under variable light conditions. In *ICT Innovations 2017. Communications in Computer and Information Science.* Springer, Cham. 19

Le, V. N. T., Ahderom, S., and Alameh, K. (2020a). Performances of the lbp based algorithm over cnn models for detecting crops andweeds with similar morphologies. *Sensors*, 20:2193. 12, 17, 20, 23, 77, 161

Le, V. N. T., Ahderom, S., Apopei, B., and Alameh, K. (2020b). A novel method for detecting morphologically similar crops and weeds based on the combination of contour masks and filtered local binary pattern operators. *GigaScience*, 9(3):giaa017. 72

Le, V. N. T., Apopei, B., and Alameh, K. (2019). Effective plant discrimination based on the combination of local binary pattern operators and multiclass support vector machine methods. *Inf. Process. Agric.*, 6:116–131. 9, 12, 14, 33

Li, N., Grift, T. E., Yuan, T., Zhang, C., Momin, M. A., and Li, W. (2016). Image processing for crop/weed discrimination in fields with high weed pressure. In *2016 ASABE International Meeting.* American Society of Agricultural and Biological Engineers. 8

Lin, F., Zhang, D., Huang, Y., Wang, X., and Chen, X. (2017). Detection of corn and weed species by the combination of spectral, shape and textural features. *Sustainability*, 9:1335. 9

Liu, B. and Bruch, R. (2020). Weed detection for selective spraying: a review. *Curr Robot Rep*, 1:19–26. 7

Liu, B., Li, R., Li, H., You, G., Yan, S., and Tong, Q. (2019). Crop/weed discrimination using a field imaging spectrometer system. *Sensors*, 19:5154. 10, 15

Liu, H., Lee, S. H., and Saunders, C. (2014). Developing and a machine vision system for weed detection during both of off-season and in-season in broadacre no.tillage cropping lands. *American Journal of Agricultural and Biological Sciences*, 9(2):174–193. 8

Liu, H., Sun, H., Li, M., and Iida, M. (2020). Application of color featuring and deep learning in maize plant detection. *Remote Sens.*, 12:2229. 9, 25, 120, 162

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 10012–10022. 17 August 2021, Venice, Italy. 99

López-Correa, J. M., Moreno, H., Ribeiro, A., and Andújar, D. (2022). Intelligent weed management based on object detection neural networks in tomato crops. *Agronomy*, 12:2953. 25

Lottes, P. (2021). *Plant classification systems for agricultural robots.* PhD thesis, University on Bonn. 43

Lottes, P., Behley, J., Chebrolu, N., Milioto, A., and Stachniss, C. (2020). Robust joint stem detection and crop-weed classification using image sequences for plant-specific treatment in precision farming. *J. Field Robot.*, 37:20–34. 17, 21, 163

Lottes, P., Behley, J., Milioto, A., and Stachniss, C. (2018). Fully convolutional networks with sequential information for robust crop and weed detection in precision farming. *IEEE Robotics and Automation Letters*, 3:2870–2877. 21, 163

Lottes, P., Hoeferlin, M., Sander, S., Muter, M., Schulze, P., and Stachniss, L. C. (2016). An effective classification system for separating sugar beets and weeds for precision farming applications. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5157–5163. Stockholm, Sweden. 9, 14, 163

Lottes, P., Hörferlin, M., Sander, S., and Stachniss, C. (2017). Effective vision-based classification for separating sugar beets and weeds for precision farming. *J. Field Robotics*, 34:1160–1178. 9, 14, 163

Louargant, M., Jones, G., Faroux, R., Paoli, J., Maillot, T., Gée, C., and Vallete, S. (2018). Unsupervised classification algorithm for early weed detection in row-crops by combining spatial and spectral information. *Remote sensing*, 10:761. 10

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. IEEE. 108

Ma, X., Deng, X., Qi, L., Jiang, Y., Li, H., Wang, Y., and Xing, X. (2019). Fully convolutional network for rice seedling and weed image segmentation at the seedling stage in paddy fields. *PLoS ONE*, 14:e0215676. 19, 26, 165

Madsen, S. L., Mathiassen, S. K., Dyrmann, M., Laursen, M. S., Paz, a., and Jørgensen, R. N. (2020). Open plant phenotype database of common weeds in denmark. *Remote sensing*, 12:1246. 19

Mathanker, S. K., Weckler, P. R., Taylor, R. K., and Fan, G. (2010). Adaboost and support vector machine classifiers for automatic weed control: Canola and wheat. In *2010 ASABE Annual International Meeting*, page 1008834. Pittsburgh, Pennsylvania, USA. 9

McCulloch, W. S. and Pitts, W. (1988). *Bulletin of Mathematical Biophysics 5*, chapter A logical calculus of the ideas immanent in nervous activity, pages 115–133. Reprinted in Anderson and Rosenfeld. 38

Meinen, C. and Rauber, R. (2015). Root discrimination of closely related crop and weed species using ft mir-atr spectroscopy. *Front. Plant. Sci.*, 6:765. 6

Michelicci, U. (2019). *Advanced applied deep learning: Convolutional neural networks and object detection.* Apress. 24

Midtiby, H. S., Mathiassen, S. K., Andersson, K. J., and Jørgensen, R. N. (2011). Performance evaluation of a crop/weed discriminating microsprayer. *Computer and Electronics in Agriculture*, 77:35–40. 11

Milioto, A., Lottes, P., and Stachniss, C. (2017). Real-time blob-wise sugar beets vs weeds classification for monitoring fields using convolutional neural networks. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences- International Conference on Unmanned Aerial Vehicles in Geomatics*, pages 41–48. Bonn, Germany. 21

Milioto, A., Lottes, P., and Stachniss, C. (2018). 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. Brisbane, QLD, Australia. 9, 19, 20

Monteiro, A. and Santos, S. (2022). Sustainable approach to weed management: The role of precision weed management. *Agronomy*, 12:118. 1, 2, 3

Moutik, O., Sekkat, H., Tigani, S., Chehri, A., Saadane, R., Tchakoucht, T. A., and Paul, A. (2023). Convolutional neural networks or vision transformers: Who will win the race for action recognitions in visual data? *Sensors*, 23:734. 16

Mueller, J. P. and Massaron, L. (2022). *Artificial Intelligence for Dummies.* Wiley. 30

Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction.* The MIT Press. 48, 52, 53, 56

Nikolić, N., Rizzo, D., Marraccini, E., Gotor, A. A., Mattivi, P., Saulet, P., Persichetti, A., and Masin, R. (2021). Site- and time-specific early weed control is able to reduce herbicide use in maize—a case study. *Ital. J. Agron.*, 16:1780. 3, 14

Nkemelu, D. and amd Nancy Lubalo, D. O. (2018). Deep convolutional neural network for plant seedlings classification. *arXiv*, page arXiv:1811.08404. 21

Ojala, T. and Pietikainen, M. (1999). Unsupervised texture segmentation using feature distributions. *Pattern Recognition*, 32(1):477–486. 32

Ojala, T., Pietikainen, M., and Harwood, D. (1996). Comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59. 32

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987. 12, 31, 33

Olsen, A., Konovalov, D. A., Philippa, B., Ridd, P., Wood, J. C., Johns, J., Banks, W., Girgenti, B., Kenny, O., Whinney, J., Calvert, B., Azghadi, M. R., and White, R. D. (2019). Deepweeds: A multiclass weed species image dataset for deep learning. *Sci. Rep.*, 9:2058. 19, 20, 161

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66. 9

Panneton, B., Guillaume, S., Roger, J., and Samson, G. (2010). Improved discrimination between monocotyledonous and dicotyledonous plants for weed control based on the blue-green region of ultraviolet-induced fluorescence spectra. *Appl Spectrosc.*, 64(1):30–36. 6

Pantazi, X. E., Moshou, D., and Bravo, C. (2016). Active learning system for weed species recognition based on hyperspectral sensing. *Biosyst Eng.*, 146:193–202. 5

Partel, V., Kakarla, S. C., and Ampatzidis, T. (2019). Development and evaluation of a low-cost and smart technology for precision weed management utilizing artificial intelligence. *Computer and electronics in Agriculture*, 157:339–350. 2

Peng, H., Li, Z., Zhou, Z., and Shao, Y. (2022). Weed detection in paddy field using an improved retinanet network. *Computers and Electronics in Agriculture*, 199:107179. 87

Pereira, L. A., Nakamura, R. Y., de Souza, G. F., Martins, D., and Papa, J. P. (2012). Aquatic weed automatic classification using machine learning techniques. *Computer and Electronics in Agriculture*, 87:56–63. 11, 15

Peteinatos, G. G., Reichel, P., Karouta, J., Andújar, D., and Gerhards, R. (2020). Weed identification in maize, sunflower, and potatoes with the aid of convolutional neural networks. *Remote sensing*, 12:4185. 23, 115

Peteinatos, G. G., Weis, M., Andújar, D., Ayala, V. R., and Gerhards, R. (2013). Potential use of ground-based sensor technologies for weed detection. *Pest Manag. Sci.*, 70:190–199. 10

Pgnatti, S., Casa, R., and Harfouche, A. (2016). Maize crop and weeds species detection by using uav vnir hyperpectral data. In *Proceedings of the IGARSS 2019—2019 IEEE International Geoscience and Remote Sensing Symposium*, page 7235–7238. Yokohama, Japan. 10

Picon, A., San-Emeterio, M. G., Bereciartua-Perez, A., Klukas, C., Eggers, T., and Navarra-Mestre, R. (2022). Deep learning-based segmentation of multiple species of weeds and corn crop using synthetic and real image datasets. *Computer and Electronics in Agriculture*, 194:106719. 1, 21, 24, 26, 87, 163, 165

Potena, C., Nardi, D., and Pretto, A. (2017). *Intelligent Autonomous Systems 14*, volume 531, chapter Fast and Accurate Crop and Weed Identification with Summarized Train Sets for Precision Agriculture, pages 105–121. Springer, Cham. 9

Pott, L. P., Amado, T. J. C., Schwalbert, R. A., Sebem, E., Jugulamd, M., and Ciampitti, I. A. (2020). Pre-planting weed detection based on ground field spectral data. *Pest Manag. Sci.*, 76:1173–1182. 5

Prema, P. and Murugan, D. (2016). A novel angular texture pattern (atp) extraction method for crop and weed discrimination using curvelet transformation. *Electronic Letters on Computer Vision and Image Analysis*, 15(1):27–59. 9

Pulido, C., Solaque, L., and Velasco, N. (2017). Weed recognition by svm texture feature classification in outdoor vegetable crop images. *Ing. Investig.*, 37:68–74. 14

Pulido-Rojas, C. A., Molina-Villa, M. A., and Solaque-Guzmán, L. E. (2016a). Machine vision system for weed detection using image filtering in vegetables crops. *Revista Facultad de Ingeniería*, 80:124–130. 8

Pulido-Rojas, C. A., Solaque-Guzman, L. E., and Velasco-Toledo, N. F. (2016b). A comparative analysis of weed images classification approaches in vegetables crops. *Engineering Journal*, 21(2):81–98. 14, 15

Pérez-Ortiz, M., Peña, J. M., Gutiérrez, P. A., and J. Torres-Sánchez, C. H. (2015). A semi-supervised system for weed mapping in sunflower crops using unmanned aerial vehicles and a crop row detection method. *Applied Soft Computing*, 37:553–544. 6, 14

Quan, L., Wu, B., Mao, S., Yang, C., and Li, H. (2021). An instance segmentation-based method to obtain the leaf age and plant centre of weeds in complex field environments. *Sensors*, 21:3389. 2, 87, 163

Rabab, S., Breen, E., Gebremedhin, A., Shi, F., Badenhorst, P., Chen, Y. P., and Daetwyler, H. D. (2021). A new method for extracting individual plant bio-characteristics from high-resolution digital images. *Remote Sens.*, 13:1212. 1

Raja, R., Nguyen, T. T., Slaughter, D. C., and Fennimore, S. A. (2020). Real-time weed-crop classification and localisation technique for robotic weed control in lettuce. *Biosystems Engineering*, 192:257–274. 1

Rakhmatulin, I. (2020). Artificial intelligence in weed recognition tasks. *Asian Journal of Applied Science and Technology*, 4(2):70–81. 8

Reedha, R., Dericquebourg, E., Canals, R., and Hafiane, A. (2022). Transformer neural network for weed and crop classification of high resolution uav images. *Remote Sens.*, 14:592. 20, 23, 99

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv*, page arXiv:1506.01497. 56

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *MICCAI 2015. Lecture Notes in Computer Science*. 99, 163

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408. 37

Sa, I., Chen, Z., Popovic, M., Khanna, R., Liebisch, F., Nieto, J., and Siegwart, R. (2018). weednet: Dense semantic weed classification using multispectral images and mav for smart farming. *IEEE Robotics and Automation Letters*, 3(1):588–595. 19

Sabottke, C. F. and Spiele, B. M. (2020). The effect of image resolution on deep learning in radiography. *Radiology: Artificial Intelligence*, 2:e190015. 20

Sabzi, S., Abbaspour-Gilandeh, Y., and Arribas, J. I. (2020). An automatic visible-range video weed detection, segmentation and classification prototype in potato fiel. *Heliyon*, 6:e03685. 2

Sahin, H. M., Miftahushudur, T., Grieve, B., and Yin, H. (2023). Segmentation of weeds and crops using multispectral imaging and crf-enhanced u-net. *Computer and Electronics in Agriculture*, 211:107956. 20, 163

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520. 94, 95, 132

Santosh, K., Das, N., and Ghosh, S. (2022). *Deep Learning Models for Medical Imaging*, chapter Deep learning: a review, pages 29–63. Academic Press. 22

Sapkota, B. B., Hu, C., and Bagavathiannan, M. V. (2022). Evaluating cross-applicability of weed detection models across different crops in similar production environments. *Front. Plant. Sci.*, 13:837726. 25

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *J. Big Data*, 6:60. 18

Siddiqi, M. H., Lee, S., and Me, A. (2014). Developing and a machine vision system for weed detection during both of off-season and in-season in broadacre no.tillage cropping lands. *Journal of Information Science and Engineering*, 30:1253–1270. 8

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *In Proceedings of the 3rd International Conference on Learning Representations*. 74, 77, 163

Skansi, S. (2018). *Introduction to Deep Learning from Logical Calculus to Artificial Intelligence*. Springer. 22, 56

Skovsen, S., Dyrmann, M., Mortensen, A. K., Laursen, M. S., Gislum, R., Eriksen, J., Farkhani, S., Karstoft, H., and Jorgensen, R. N. (2019). The grassclover image dataset for semantic and hierarchical species understanding in agriculture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 19

Söderkvist, O. J. O. (2001). *Computer vision classification of leaves from Swedish trees*. PhD thesis, Linkoping University. 160

Soille, P. (2004). *Morphological Image Analysis*, chapter Opening and Closing. Springer, Berlin, Heidelberg. 20

Soylu, B. E., Guzel, M. S., Bostanci, G. E., Ekinci, F., Asuroglu, T., and Acici, K. (2023). Deep-learning-based approaches for semantic segmentation of natural scene images: A review. *Electronics*, 12:2730. 24

Steward, B., Gai, J., and Tang, L. (2019). *Robotics and automation for improving agriculture*, chapter The use of agricultural robots in weed management and control. Burleigh Dodds Science Publishing. 4

Strudel, R., Garcia, R., Laptev, I., and Schmid, C. (2021). Segmenter: Transformer for semantic segmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7242–7252. Montreal, QC, Canada. 99

Subeesh, A., Bhole, S., Singh, K., Chandel, N., Rajwade, Y., Rao, K., Kumar, S., and Jat, D. (2022). Deep convolutional neural network models for weed detection in polyhouse grown bell peppers. *Artificial Intelligence in Agriculture*, 6:47–54. 21, 23

Sudars, K., Jasko, J., Namatevs, I., Ozola, L., and Badaukis, N. (2020). Dataset of annotated food crops and weed images for robotic computer vision control. *data in Brief*, 31:105833. 19

Suh, H. K., IJsselmuiden, J., Hofstee, J. W., and van Henten, E. J. (2018). Transfer learning for the classification of sugar beet and volunteer potato under field conditions. *Biosystems Engineering*, 174:50–65. 23

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence.* 78

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 78

Tang, J., Chen, X., Miao, R., and Wang, D. (2016). Weed detection using image processing under different illumination for site-specific areas spraying. *Computer and Electronics in Agriculture*, 122:103–111. 13, 14, 20

Taulli, T. (2019). *Artificial Intelligence Basics: A Non-Technical Introduction.* Apress. 30

Teimouri, N., Dyrmann, M., Nielsen, P. R., Mathiassen, S. K., Somerville, G. J., and Jørgensen, R. N. (2018). Weed growth stage estimator using deep convolutional neural networks. *Sensors*, 18:1580. 19

Tellaeche, A., Burgos-Artizzu, X. P., Pajares, G., and Ribeiro, A. (2008). A vision-based method for weeds identification through the bayesian decision theory. *Pattern Recognition*, 41:521–530. 13

Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., and Chavez-Urbiola, E. A. (2023). Loss functions and metrics in deep learning. *arXiv preprint arXiv:2307.02694v2*. 44, 61

Thambawita, V., Strümke, I., Hicks, S. A., Halvorsen, P., Parasa, S., and Riegler, M. A. (2021). Impact of image resolution on deep learning performance in endoscopy image classification: An experimental study using a large dataset of endoscopic images. *Diagnostics*, 11:2183. 20

Theckedath, D. and Sedamkar, R. R. (2020). Detecting affect states using vgg16, resnet50 and se-resnet50 networks. *SN Comput. Sci.*, 1:79. 77

Torres-Sánchez, J., Mesas-Carrascosa, F. J., Jiménez-Brenes, F. M., de Castro, A. I., and López-Granados, F. (2021). Early detection of broad-leaved and grass weeds in wide row crops using artificial neural networks and uav imagery. *Agronomy*, 2021:749. 14

Vaidhehi, M. and Malathy, C. (2022). An unique model for weed and paddy detection using regional convolutional neural networks. *ACTA AGRICULTURAE SCANDINAVICA, SECTION B–SOIL & PLANT SCIENCE*, 72:463–475. 21

Van Der Weide , R. Y., Bleeker, P. O., Achten, V. T. J. M., Lotz, L. A. P., Fogelberg, F., and Melander, B. (2008). Innovation in mechanical weed control in crop rows. *Weed Research*, 48:215–224. 2

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems*, pages 6000–6010. 6 December, Long Beach, California, USA. 16, 56, 57, 99, 103

Vrindts, E., De Baerdemaeker, J., and Ramon, H. (2002). Weed detection using canopy reflection. *Precision Agriculture*, 3:63–80. 4

Wang, A., Xu, Y., Wei, X., and Cui, B. (2020). Semantic segmentation of crop and weed using an encoder-decoder network and image enhancement method under uncontrolled outdoor illumination. *IEEE Acess*, 8:81724–81734. 8, 9, 19

Wang, A., Zhang, W., and Wei, X. (2019). A review on weed detection using ground-based machine vision and image processing techniques. *Computer and Electronics in Agriculture*, 158:226–240. 2, 4, 8, 10, 12

Wang, H., Liu, W., Zhao, K., Yu, H., Zhang, J., and Wang, J. (2018). Evaluation of weed control efficacy and crop safety of the new hppd-inhibiting herbicide-qyr301. *Sci. Rep.*, 8:7910. 2

Wang, Y., Zhang, S., Dai, B., Yang, S., and Song, H. (2023). Fine-grained weed recognition using swin transformer and two-stage transfer learning. *Front. Plant Sci.*, 14:1134932. 20, 21, 23, 99

weis, M. and Sökefeld, M. (2010). Detection and identification of weeds. In *Precision Crop Protection - the Challenge and Use of Heterogeneity*. Springer, Netherlands, Dordrecht. 7

Wessner, R. N., Frozza, R., Duarte da Silva Bagatini, D., and Molz, R. F. (2023). Recognition of weeds in corn crops: System with convolutional neural networks. *Journal of Agriculture and Food Research*, 14:100669. 23

Westwood, J. H., Charudattan, R., Duke, S. O., Fennimore, S. A., Marrone, P., Slaughter, D. C., Swanton, C., and Zollinger, R. (2018). Weed management in 2050: Perspectives on the future of weed science. *Weed Sci.*, 66:275–285. 1

Wu, S. G., Bao, F. S., Xu, E. Y., Wang, Y.-X., Chang, Y.-F., and Xiang, Q.-L. (2007). A leaf recognition algorithm for plant classification using probabilistic neural network. In *IEEE International Symposium on Signal Processing and Information Technology*. 160

Wu, X., Xu, W., Song, Y., and Cai, M. (2011). A detection method of weed in wheat field on machine vision. *Procedia Eng.*, 15:1998–2003. 9, 13

Wu, Z., Chen, Y., Zhao, B., Kang, X., and Ding, Y. (2021). Review of weed detection methods based on computer vision. *Sensors*, 21:3647. 11, 12, 13, 15

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Álvarez, J. M., and Luo, P. (2021). Segformer: Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems*. 99, 102

Xu, B., Fan, J., Chao, J., Arsenijevic, N., Werle, R., and Zhang, Z. (2023). Instance segmentation method for weed detection using uav imagery in soybean fields. *Computer and Electronics in Agriculture*, 211:107994. 17, 26, 99

Xu, K., Li, H., Cao, W., Zhu, Y., Chen, R., and Ni, J. (2020a). Recognition of weeds in wheat fields based on the fusion of rgb images and depth images. *IEEE Access*, 8:110362–110370. 14

Xu, Y., Gao, Z., Khot, L., Meng, X., and Zhang, Q. (2018). A real-time weed mapping and precision herbicide spraying system for row crops. *Sensors*, 18:4245. 13

Xu, Y., He, R., Gao, Z., Li, C., Zhai, Y., and Jiao, Y. (2020b). Weed density detection method based on absolute feature corner points in field. *Agronomy*, 10:113. 12, 13

Yang, W., Wang, S., Zhao, X., Zhang, J., and Feng, J. (2015). Greenness identification based on hsv decision tree. *INFORMATION PROCESSING IN AGRICULTURE*, 2:149–160. 8, 9, 70

Yeganehpoor, F. S. (2015). Effects of cover crops and weed management on corn yield. *Joural of the Saudi Society of Agricultural Sciences.*, 14(2):178–181. 2

Yu, J., Schumann, A. W., Cao, Z., Sharpe, S. M., and Boyd, N. S. (2019a). Weed detection in perennial ryegrass with deep learning convolutional neural network. *Front. Plant. Sci.*, 10:1422. 19

Yu, J., Sharpe, S. M., Schumann, A. W., and Boyd, N. S. (2019b). Detection of broadleaf weeds growing in turfgrass with convolutional neural networks. *Pest Manag Sci*, 75:2211–2218. 161

Zenkl, R., Timofte, R., Kirchgessner, N., Roth, L., Hund, A., Gool, L. V., Walter, A., and Aasen, H. (2022). Outdoor plant segmentation with deep learning for high-throughput field phenotyping on a diverse wheat dataset. *Front. Plant. Sci.*, 12:774068. 26, 87, 165

Zhang, J., Gong, J., Zhang, Y., Mostafa, K., and Yuan, G. (2023). Weed identification in maize fields based on improved swin-unet. *Agronomy*, 13:1846. 17, 20, 21, 24, 27

Zhang, J., Su, W., Zhang, H., and Peng, Y. (2022). Se-yolov5x: An optimized model based on transfer learning and visual attention mechanism for identifying and localizing weeds and vegetables. *Agronomy*, 12:2061. 25

Zhang, S., Guo, J., and Wang, Z. (2019a). Combing k-means clustering and local weighted maximum discriminant projections for weed species recognition. *Front. Comput. Sci.*, 1:4. 15

Zhang, Y., Gao, J., Cen, H., Lu, Y., Yu, X., He, Y., and Pieters, J. G. (2019b). Automated spectral feature extraction from hyperspectral images to differentiate weedy rice and barnyard grass from a rice crop. *Computer and Electronics in Agriculture*, 159:42–49. 10

Zhang, Y. and He, Y. (2006). Crop/weed discrimination using near-infrared reflectance spectroscopy (nirs). In *Proceedings of the Fourth International Conference on Photonics and Imaging in Biology and Medicine*, page 60472G. Tianjin, China. 6

Zhou, Q., Huang, Z., Zheng, S., Jiao, L., Wang, L., and Wang, R. (2022). A wheat spike detection method based on transformer. *Front. Plant Sci.*, 13:1023924. 99

Zhu, D. S., Pan, J. Z., and He, Y. (2008). Identification methods of crop and weeds based on vis/nir spectroscopy and rbf-nn model. In *PMID*, page 1102–1106S. China. 6, 14

Zou, K., Liao, Q., Zhang, F., Che, X., and Zhang, C. (2022). A segmentation network for smart weed management in wheat fields. *Computer and Electronics in Agriculture*, 202:107306. 20, 21, 26

Zwiggelaar, R. (1998). A review of spectral properties of plants and their potential use for crop/weed discrimination in row-crops. *Crop Prot.*, 17:189–206. 4

A

Appendix A

A.1. Code for calibration

Algorithm 1 Computation of the centers of the sprayer nozzles, split image coordinates and spray factor

Input: Calibration RGB image I_{RGB}

Output: $NC_N \in \mathbb{R}^6$, $NC_B \in \mathbb{R}^6$ Center coordinates for NLW and BLW nozzles; $SI \in \mathbb{R}^7$ Coordinates to split the predicted image; $Sf_N \in \mathbb{R}^6$, $Sf_B \in \mathbb{R}^6$ Spray factors for NLW and BLW nozzles

```
  Step 1 Thresholding and binarization
  $I_{HSV} \leftarrow$ Color space transformation from $I_{RGB}$ to $I_{HSV}$
  $t \leftarrow Thresholding(I_{HSV})$                ▷ Thresholding using Equation 3.1
  $I_b \leftarrow Binarization(I_{HSV}, t)$           ▷ $I_{HSV}$ binarization according to $t$

  Step 2 Centroid coordinates estimation
  $C \leftarrow ContourExtraction(I_b)$               ▷ Contours extraction using CCA
  $x = \emptyset$
  $y = \emptyset$
```

Algorithm 1 Continue.

```
for $i$ in range $len(C)$ do
    $a \leftarrow ContourArea(C_i)$                ▷ Pixel area computation of $C_i$
    if $a \geq 400$ then
        $M \leftarrow Moments(C_i)$                ▷ Compute the moments of $C_i$
        $x \leftarrow append(M_x)$
        $y \leftarrow append(M_y)$
    end if
end for

Step 3 Center coordinates of NLW and BLW nozzles
$y_s \leftarrow sort(y)$
$yN \leftarrow y_s[0:6]$
$yB \leftarrow y_s[6:len(y_s)]$
$NC_N = \emptyset$
for $i$ in $yN$ do
    $xN_i \leftarrow$ The coordinate in $x$ list according to $i$ position in $y$ list
    $NC_N \leftarrow append(xN_i, YN_i)$
end for
$NC_B = \emptyset$
for $i$ in $yB$ do
    $xB_i \leftarrow$ The coordinate in $x$ list according to $i$ position in $y$ list
    $NC_B \leftarrow append(xB_i, YB_i)$
end for
$NC_N \leftarrow$ Sort coordinates for NLW nozzles concerning the abscissas
$NC_B \leftarrow$ Sort coordinates for BLW nozzles concerning the abscissas

Step 4 Computation of coordinates to split predicted image and spray factors
$H \leftarrow$ Height of $I_{RGB}$
$xN \leftarrow$ Abscissas in $NC_N$
$yN \leftarrow$ Ordinates in $NC_N$
$xB \leftarrow$ Abscissas in $NC_B$
$yB \leftarrow$ Ordinates in $NC_B$
```

Algorithm 1 Continue.

```
$SI = \emptyset$
$Sf_N = \emptyset$
$Sf_B = \emptyset$
for $i$ in range $(len(xN) - 1)$ do
    Step 4.1 Computation spray factors for NLW and BLW nozzles
    $Sf_{Ni} \leftarrow append((xN[i+1] - xN[i])/4)$    ▷ Spray factor for NLW nozzles
    $Sf_{Bi} \leftarrow append((xB[i+1] - xB[i])/4)$    ▷ Spray factor for BLW nozzles
    Step 4.2 Computation coordinates to split predicted image
    $aN_i \leftarrow (xN[i+1] - xN[i])/2 + xN[i]$    ▷ Midpoint among two consecutive abscissas
    $oN_i \leftarrow (yN[i+1] + yN[i])/2$ ▷ The mean among two consecutive ordinates
    $aB_i \leftarrow (xB[i+1] - xB[i])/2 + xB[i]$    ▷ Midpoint among two consecutive abscissas
    $oB_i \leftarrow (yB[i+1] + yB[i])/2$ ▷ The mean among two consecutive ordinates
    $m_i \leftarrow$ Slope computation using $m = \frac{y_2 - y_1}{x_2 - x_1}$
    $ytop_i = 0$    ▷ Line starts in $H = 0$
    $xtop_i \leftarrow$ Computation using point-slope equation form of a line
    $ybtn_i = H$    ▷ Line ends in image height
    $xbtn_i \leftarrow$ Computation using point-slope equation form of a line
    $SI \leftarrow append([(xtop_i, ytop_i), (xbtn_i, ybtn_i)])$
    if $i = 0$ then
        $aN_i \leftarrow xN[i] - (xN[i+1] - xN[i])/2$
        $oN_i \leftarrow (yN[i+1] + yN[i])/2$
        $aB_i \leftarrow xB[i] - (xB[i+1] - xB[i])/2$
        $oB_i \leftarrow (yB[i+1] + yB[i])/2$
        $m_i \leftarrow$ Slope computation
        $ytop_i = 0$    ▷ Line starts in $H = 0$
        $xtop_i \leftarrow$ Computation using point-slope equation form of a line
        $ybtn_i = H$    ▷ Line ends in image height
        $xbtn_i \leftarrow$ Computation using point-slope equation form of a line
        $SI \leftarrow append([(xtop_i, ytop_i), (xbtn_i, ybtn_i)])$ ▷ Append in index 0 of $SI$
    end if
end for
```

Algorithm 1 Continue.

```
if $i = len(xN) - 1$ then
    Step 4.1 Continue...                                    ▷ Spray factor for the last nozzles
    $Sf_{Ni} \leftarrow append((xN[i+1] - xN[i])/4)$
    $Sf_{Bi} \leftarrow append((xB[i+1] - xB[i])/4)$
    Step 4.2 Continue...
    $aN_i \leftarrow xN[i+1] + (xN[i+1] - xN[i])/2$
    $oN_i \leftarrow (yN[i+1] + yN[i])/2$
    $aB_i \leftarrow xB[i+1] - (xB[i+1] - xB[i])/2$
    $oB_i \leftarrow (yB[i+1] + yB[i])/2$
    $m_i \leftarrow$ Slope computation
    $ytop_i = 0$                                            ▷ Line starts in $H = 0$
    $xtop_i \leftarrow$ Computation using point-slope equation form of a line
    $ybtn_i = H$                                            ▷ Line ends in image height
    $xbtn_i \leftarrow$ Computation using point-slope equation form of a line
    $SI \leftarrow append([(xtop_i, ytop_i), (xbtn_i, ybtn_i)])$   ▷ Append in last index of $SI$
end if
```

A.2. Smart spraying algorithm

Algorithm 2 Smart spraying algorithm under authentic cornfields

```
Input: RGB image captured in authentic cornfield $I_{RGB}$; Center coordinates of
nozzles $NC_N$ and $NC_B$; Coordinates to split predicted images $SI \in \mathbb{R}^7$; Spray
factors $Sf_N \in \mathbb{R}^6$ and $Sf_B \in \mathbb{R}^6$ for NLW and BLW nozzles
Output: $SP_{NLW} \in \mathbb{R}^6$; $SP_{BLW} \in \mathbb{R}^6$    ▷ Binary signals $[1, 0]$ to open or close
each nozzle
  $yN \leftarrow$ Ordinates in $NC_N$
  $yB \leftarrow$ Ordinates in $NC_B$
  $Sf_N \leftarrow$ Spray factors for NLW nozzles
  $Sf_B \leftarrow$ Spray factors for BLW nozzles

  Step 1 Smart segmentation of input image
  $I_S \leftarrow$ Segmentation of $I_{RGB}$ using SegFormer model

  Step 2 Extraction of NLW and BLW masks
  $P_{NLW} = \emptyset$                              ▷ List of the predicted NLW weeds
  $P_{BLW} = \emptyset$                              ▷ List of the predicted BLW weeds
  for $i$ in range len(6) do
      $R_i \leftarrow$ The $ith$ split region of the predicted image using respective coordi-
  nates in $SI$
      $Cat_i \leftarrow$ Categorizing $R_i$ with respect the number of classes
      $P_{NLW} \leftarrow append(mask, NLW)$         ▷ Append the NLW prediction mask
  region
      $P_{BLW} \leftarrow append(mask, BLW)$ ▷ Append the BLW prediction mask region
  end for
```

Algorithm 2 Continue.

```
Step 3 Ordinates of each region inside each NLW mask slice
$Y_{NLW} = \emptyset$
for $mask\ in\ P_{NLW}$ do
    $C \leftarrow ContourExtraction(mask)$
    $y_{pred} = \emptyset$                          ▷ Ordinates of regions inside $mask_i$
    for $i\ in\ rage\ len(C)$ do
        $a \leftarrow ContourArea(C_i)$             ▷ Pixel area computation of contour $i$
        if $a \geq 150$ then
            $I_{RGB} \leftarrow BoundRect(C_i)$   ▷ Bounding box around contour $i$ in $I_{RGB}$
            $M \leftarrow Moments(C_i)$           ▷ Compute the moments of contour $i$
            $My_i \leftarrow$ Ordinate of centroid of contour $i$
            $y_{pred} \leftarrow append(My_i)$
        end if
    end for
    $Y_{NLW} \leftarrow append(y_{pred})$
end for

Step 4 Ordinates of each region inside each BLW mask slice
$Y_{BLW} = \emptyset$              ▷ Ordinates of each region inside each BLW mask slice
for $mask\ in\ P_{BLW}$ do
    $C \leftarrow ContourExtraction(mask)$
    $y_{pred} = \emptyset$                          ▷ Ordinates of regions inside $mask_i$
    for $i\ in\ rage\ len(C)$ do
        $a \leftarrow ContourArea(C_i)$             ▷ Pixel area computation of contour $i$
        if $a \geq 300$ then
            $I_{RGB} \leftarrow BoundRect(C_i)$   ▷ Bounding box around contour $i$ in $I_{RGB}$
            $M \leftarrow Moments(C_i)$           ▷ Compute the moments of contour $i$
            $My_i \leftarrow$ Ordinate of centroid of contour $i$
            $y_{pred} \leftarrow append(My_i)$
        end if
    end for
    $Y_{BLW} \leftarrow append(y_{pred})$
end for
```

Algorithm 2 Continue.

```
Step 5 Deriving the $SP_{NLW}$ and $SP_{BLW}$ vectors for turning “On” or “Off” the nozzles
$SP = \emptyset$                                   ▷ $SP$ could be either $SP_{NLW}$ or $SP_{BLW}$
for $i$ in range $len(Y)$ do                       ▷ $Y$ could be either $Y_{NLW}$ or $Y_{BLW}$
    $ord = sort(Y_i)$          ▷ $ord$ is the $i$ list of ordinates in either $Y_{NLW}$ or $Y_{BLW}$
    $y_{nozz} = y_i$           ▷ $y_{nozz}$ is the $i$ ordinate of nozzles in either $yN$ or $yB$
    $Sf = Sf_i$                ▷ $Sf$ is the $i$ spray factor for the $i$ in either $Sf_N$ or $Sf_B$
    $close = \emptyset$
    $ordact \leftarrow$ Actualized $ord$ list with solely the $y_i \geq y_{nozz}$
    for $j$ in rage $len(ordact)$ do
        if $ordact[j] - ordact[0] \leq Sf$ then
            $close \leftarrow append(ordact[j])$
        end if
    end for
    if $len(close)! = 0$ then
        $y_{fore} = mean(close)$
        $S \leftarrow y_{fore} - y_{nozz}$
        if $0 \leq S \leq Sf$ then                              ▷ Spray condition
            $SP \leftarrow append(1)$
        else
            $SP \leftarrow append(0)$
        end if
    else
        $SP \leftarrow append(0)$
    end if
end for
```