POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Water Stress Detection in Potato Crops Using Multispectral Imaging and Advanced Object Detection Models

Supervisors

Prof. Renato FERRERO

Dr. Nicola DILILLO

Candidate

Sarina TAKALLOO

October 2024

Abstract

In recent years, the agricultural sector has been rapidly transformed by the use of advanced technologies, often referred to as 'smart farming' or 'smart agriculture'. Digital tools and other scientific and technological developments have been widely used to revolutionise agricultural practices for better productivity and sustainability. Detecting water stress in plants is one of these challenges to be addressed. Typically, soil moisture sensors are used to assess the condition of the crop. However, the state of the art has identified multispectral imagery as a promising method for detecting water stress in crops, especially using near-infrared (NIR) and red-edge bands. This study investigates the use of multispectral imagery that doesn't only use RGB channels but also NIR and Red-Edge channels to improve the detection of water stress in potato crops using advanced object detection models. This dataset contains two classes: stressed and healthy, for both RGB and spectral images (Red, Green, Near-Infrared, and Red-Edge). There are 300 images used for training and 60 for testing in both the RGB and each spectral band. Additionally, an augmented dataset is provided, consisting of 1500 training images and 60 testing images for both RGB and spectral data. Multispectral imaging, especially incorporating NIR and Red-Edge bands, offers several advantages over traditional plant sensors, including non-invasive, remote assessment capabilities and the ability to cover large areas more efficiently. These imaging techniques provide valuable insights into plant health by capturing data beyond the visible spectrum, particularly in identifying early signs of water stress. Building on this potential, several studies have been carried out. The first was to investigate the performance of the YOLOv8 model with 4-channel inputs (Red + Green + NIR + Red Edge) and RGBN (RGB + NDVI) on a potato dataset, evaluating their behaviour in comparison with traditional RGB images. The second was conducted to explore the potential of combining multiple channels, specifically RGN (Red, Green, NIR) and RGE (Red, Green, Red Edge), which were used to train a YOLOv8 model that was compared to the same model trained using RGB images. In addition to YOLOv8, a Faster R-CNN model with a ResNet50 backbone was trained and evaluated on the multispectral image configurations for comparative analysis. Finally, a pre-trained YOLOv8 model from Hugging Face, specifically designed for plant leaf detection and classification, was used for additional investigation and benchmarking. In all of this work, both cross-validation and traditional validation approaches were used to assess model performance. The results of this study show that the combination of multispectral data significantly improves detection accuracy, confirming that the integration of these spectral bands can improve the identification of water-stressed plants compared to RGB-only approaches.

Summary

Introduction

Detecting water stress in plants is a crucial challenge to address in agricultural practices. Typically, soil moisture sensors are used to assess the condition of the crop. However, recent advancements have identified multispectral imagery as a promising method for detecting water stress, particularly using near-infrared (NIR) and red-edge bands. This study investigates the use of multispectral imagery with RGB, NIR, and red-edge channels to improve the detection of water stress in potato crops using advanced object detection models.

Multispectral imaging, especially when incorporating NIR and Red-Edge bands, provides several advantages over traditional plant sensors, including non-invasive, remote assessment capabilities and the ability to cover large areas efficiently. These imaging techniques capture valuable insights into plant health by collecting data beyond the visible spectrum, which is particularly effective in identifying early signs of water stress.

Objective

This study investigates two main objectives:

- 1) **Evaluating the Presence of Spectral Bands with Respect to RGB Images Using YOLO and Faster R-CNN:** This research seeks to assess how the inclusion of spectral bands, such as Near-Infrared (NIR) and Red-Edge, improves the detection performance of models like YOLOv8 and Faster R-CNN compared to RGB images alone. The goal is to understand the impact of these spectral bands on the accuracy and effectiveness of object detection models in identifying water-stressed plants.
- 2) **Evaluating the Presence of an Extra Dimension Using YOLO with 4 Channels:** This question investigates whether combining multiple spectral band leads to better detection performance when using YOLOv8

with four-channel inputs. By adding this extra dimension, the study aims to determine if it enhances the model's ability to detect stress compared to traditional RGB inputs.

Methodology

A dataset from the University of Idaho was utilized, featuring high-resolution RGB and multispectral images captured via a Parrot Sequoia camera. RGB images were 3456 × 4608 pixels, while monochrome sensors captured images at 1280 × 960 pixels for red, green, red-edge, and NIR channels. The dataset contained 360 RGB images, split into 300 for training and 60 for testing, along with augmented versions to increase diversity. Multispectral images (416 × 416 pixels) were also used for different spectral band combinations, specifically for detecting crop health under various water stress conditions. To facilitate model training, annotations were converted to the YOLO format, normalized, and organized into train, validation, and test sets. Image augmentation techniques, such as blurring, grayscale transformation, CLAHE, and mosaic augmentation, were applied to enhance the training dataset. Furthermore, the RGB and multispectral images were merged into different configurations to form input combinations: RGB, RGN (Red, Green, Near-Infrared), RGE (Red, Green, Red-Edge), RGBN (RGB +NDVI) and RGREN (Red, Green, Red-Edge, Near-Infrared). For the 4-channel RGREN and RGBN setup, specific modifications were applied to the YOLOv8 architecture, allowing the model to fully utilize the color and spectral richness offered by these bands. The YOLOv8 model was deployed with two variants, YOLOv8n and YOLOv8s, with the aim of optimizing hyperparameters such as learning rates (0.1, 0.01, 0.001) and batch sizes (16, 32). The best-performing configuration from YOLOv8n was used for YOLOv8s to avoid overfitting. Faster R-CNN was also implemented, primarily focusing on RGN and RGE image combinations. Given its high computational demand, Faster R-CNN was run with a batch size of 32 over 100 epochs, using Stochastic Gradient Descent (SGD) with a learning rate of 0.001. Each model configuration was selected to leverage specific spectral properties of the bands used. NIR and Red-Edge channels were particularly effective in detecting water stress due to their sensitivity to plant health, making these spectral properties essential for optimizing both YOLOv8 and Faster R-CNN in this application.

Results and Discussions

Below all the observed improvements of multispectral images comparison with RGB is listed:

RGB vs Multispectral

Configuration	Class	Metric	Configuration Value	RGB Value
RGN	Stressed	mAP50	0.924	0.922
RGE	Healthy	Recall	0.900	0.892
RGE	Healthy	mAP50	0.952	0.940
RGE	Stressed	Recall	0.910	0.858
RGE	Stressed	mAP50	0.950	0.922
RGBN	Healthy	Recall	0.905	0.892
RGBN	Healthy	mAP50	0.958	0.940
RGBN	Stressed	Recall	0.910	0.858
RGBN	Stressed	mAP50	0.953	0.922
RGBN	Stressed	mAP50-95	0.728	0.711

Table 1: Comparison of Configurations with RGB Values

Using Faster R-CNN and a pretrained YOLOv8s model, additional improvement was observed:

Pretrained Hugging Face Model: With RGN data, this model significantly improved precision and recall. In the Healthy class, it reached 0.972 in precision and 0.959 in recall, which outperformed models such as Retina-UNet-Ag, known for lower precision and recall in both Healthy and Stressed classes. These results indicate that advanced models trained on multispectral data effectively capture detailed features, enhance generalization, and improve the distinction between healthy and stressed vegetation.

Faster R-CNN: In the RGE configuration, Faster R-CNN showed strong performance in both precision and recall, achieving 0.880 precision and 0.954 recall for the Healthy class. This suggests that RGE data enhances the model's ability to detect subtle plant health features, providing complementary information to RGB. The advantage of RGE is particularly evident in its edge detection capabilities, which aid in distinguishing healthy from stressed plants.

RGBN vs. RGREN

YOLOv8 Models: Comparing RGBN and RGREN, it was observed that RGBN generally outperformed RGREN across both the Healthy and Stressed classes. For example, in the Healthy class, YOLOv8m with RGBN achieved 0.905 recall and 0.958 mAP50, whereas YOLOv8m with RGREN recorded lower recall and mAP50 values (0.693 and 0.818, respectively). This highlights the effectiveness of NDVI in conjunction with RGB, suggesting it better complements the YOLOv8 model architecture, likely by enhancing contrast between healthy and stressed vegetation.

Precision and Recall Trade-Offs: While RGREN showed slightly improved precision in YOLOv8n for the Stressed class (0.834), it struggled to match the

recall and mAP of RGBN. This suggests that while the addition of red-edge and near-infrared bands can boost precision in certain cases, it may not capture plant health indicators as effectively as NDVI.

Conclusion

Across the board, RGBN consistently achieved the highest recall and mAP50 scores for YOLOv8s, suggesting that this configuration provides a comprehensive view of crop health. RGN and RGE configurations offered specific advantages in advanced model architectures, such as Faster R-CNN and pretrained models, enhancing recall and fine details necessary for accurate classification. The results underscore the benefits of utilizing multispectral data, particularly NDVI, when aiming to enhance model performance on healthy vs. stressed crop detection tasks. Advanced configurations like RGREN can further improve model accuracy for specific metrics, though RGBN remains the most balanced choice for achieving both high recall and mAP, especially in YOLOv8 models optimized for agricultural applications.

Acknowledgements

I am deeply grateful to Prof. Renato Ferrero for his invaluable guidance, insightful feedback, and unwavering support throughout this research. His expertise and dedication have been instrumental in shaping this thesis.

I am also sincerely thankful to Dr. Nicola Dilillo for his mentorship, technical advice, and encouragement. His contributions have been vital to the completion of this work, and his commitment to academic excellence has continually inspired me.

I extend my deepest gratitude to my family, and especially to my husband, Dr. Abolfazl Malti, for his support and belief in me. His encouragement has provided me with the strength to persevere in this journey.

I dedicate this work to the brave souls in Iran, To the men and women who, with courage, stand, fighting for the rights and freedoms they claim, lighting the path with liberty's flame. For those who dreamed, a homeland meant a place to belong.

Donna, Vita, Libertà

Table of Contents

List of Tables

List of Figures

Acronyms

AI
Artificial Intelligence

IoT
Internet of Things

RGB
Red, Green, and Blue

NIR
Near-Infrared

NDVI
Normalized Difference Vegetation Index Internet of Things

DL
Deep Learning

ML
Machine Learning

CNN
Convolutional Neural Networks

YOLO
You Only Look Once

IoU
Intersection of Union

UAV

Unmanned Aerial Vehicles

R-CNN

Region-based Convolutional Neural Network

mAP

Mean Average Precision

TP

True Positive

FP

False Positive

TN

True Negative

FN

False Negative

Chapter 1

Introduction

Smart Agriculture, often referred to as 'smart farming,' represents a shift in agricultural practices through the integration of advanced technologies with traditional methods. The overall progress in agriculture has been closely related to great changes in productivity, which derive from and happen during different eras of scientific and technological breakthroughs [1].

1. 1.0: This period, spanning from 1784 to around 1870, was characterized by traditional farming methods reliant on human and animal labor. The primary challenge during this era was low operational efficiency.

2. 2.0: During the 20th century, agriculture entered an era of mechanization. The main challenge in this period was the ineffective utilization of resources.

3. 3.0: From 1992 to 2017, agriculture saw rapid advancements in automation. However, the primary challenge was the limited intelligence in systems.

4. 4.0: Starting in 2017 and characterized by autonomous operations, this era is defined by the integration of advanced information technologies to enhance and intelligently manage agricultural processes.

Thus, while Agriculture 1.0 relied heavily on human and animal labor, Agriculture 4.0 leverages recent developments in digital advancements, such as Unmanned Aerial Vehicles (UAVs) and the Internet of Things (IoT) [2], to revolutionize farming. These innovations are a paradigm shift that allows for the more efficient use of land, thereby increasing productivity and promoting sustainability [3]. The transition from traditional to smart farming has brought forth new tools for precision agriculture, such as multispectral imaging, which enables farmers to monitor crop health and detect water stress through non-invasive methods. This technological shift enhances decision-making and allows large-scale, real-time monitoring of farmland.

Unlike traditional soil moisture sensors, which provide localized data, multispectral imaging captures information over large areas and across different spectral bands, such as near-infrared (NIR) and red-edge. These bands are especially crucial for assessing plant health, as they can provide early indicators of water stress, such as changes in chlorophyll content and photosynthesis rates [4]. As a result, multispectral imaging offers a more comprehensive approach to monitoring crop health and is a key tool in smart farming.

Moreover, smart agriculture addresses critical economic challenges, including the need to meet the growing global food demand. By 2050, the global population is expected to reach 9.73 billion, presenting a significant threat to food security, as predicted by the Food and Agriculture Organization (FAO) [2]. With this increasing demand comes the necessity for more efficient agricultural practices, particularly in the management of vital resources like water.

Water management, in particular, is becoming a prominent challenge in agriculture, especially in regions prone to water stress. As the global population rises, so too does the demand for food, which places immense pressure on agricultural systems to make efficient use of water. Water stress, which occurs when plants do not receive sufficient water for growth, is a significant barrier to agricultural productivity. As such, this issue has driven extensive research into technologies that can address water stress in smart agriculture [5].

Alongside economic concerns, smart agriculture also seeks to promote environmental sustainability, especially in the context of food production. Traditional agricultural practices have often contributed to environmental degradation, wasting water and mismanaging resources. In contrast, technologies like artificial intelligence (AI) and IoT offer innovative solutions for early detection of water stress by providing precise data on soil features, moisture levels, and climatic conditions [2]. By harnessing these technologies, farmers can make informed, real-time decisions that optimize resource use and reduce waste. Such technological advancements play a crucial role in mitigating environmental degradation and promoting long-term sustainability.

Imaging technologies have become a key enabler of smart agriculture, particularly in tasks like monitoring plant health, detecting stress, and optimizing resource management. Image processing, which has been widely used in agriculture, allows for early detection of plant stress through visual cues, enabling timely intervention. This technology, initially developed for satellite imagery and medical diagnoses, has been adapted for agricultural contexts where it is essential for processing large quantities of visual data [6]. In particular, multispectral imagery has proven to be a valuable asset for monitoring crops, offering a non-invasive way to assess plant health by capturing data outside the visible spectrum. This study focuses on the use of NIR and red-edge bands to detect water stress in potato crops, a critical issue in modern agriculture.

Object detection, a key component of image processing in agriculture, is essential for identifying specific elements within images. Deep learning models such as YOLO (You Only Look Once) and Faster R-CNN are widely used in agricultural contexts for their high accuracy and real-time performance [7, 8]. These models are particularly useful for large-scale operations, enabling tasks such as plant stress detection, weed control, and disease mapping. In addition, segmentation models like U-Net allow for pixel-wise classification of images, further enhancing the ability to differentiate between crops, weeds, and soil [9].

Deep learning models, especially Convolutional Neural Networks (CNNs), have become integral to modern agricultural image processing. CNNs excel at identifying spatial features in images, making them well-suited for tasks such as stress detection, disease identification, and yield prediction [10, 11]. More advanced architectures like ResNet and DenseNet have further enhanced CNN performance, enabling deeper networks and improved feature extraction [10]. These models are highly effective in analyzing both RGB and multispectral images, making them indispensable in agricultural monitoring.

In addition to CNNs, transformer-based models, such as Vision Transformers (ViTs), are gaining traction in agricultural applications. Originally developed for natural language processing, ViTs have shown promise in image classification and segmentation tasks, particularly when working with large-scale satellite or UAV imagery [12]. These models provide a more global perspective compared to traditional CNNs by processing images as sequences of patches, capturing long-range dependencies that improve classification accuracy [13].

The combination of advanced image processing techniques and deep learning models has opened new frontiers in smart agriculture. Precision agriculture, which relies on remote sensing, UAVs, and IoT sensors, has greatly benefited from these advancements [14, 15]. By integrating these technologies, farmers can monitor crop health, predict yields, and make data-driven decisions that enhance resource efficiency and improve sustainability.

IoT plays a pivotal role in the modern agricultural landscape, enabling real-time data collection and analysis. Multispectral sensors, in particular, have become critical tools for assessing plant health by capturing data in different light bands, such as NIR and red-edge. These sensors can detect early signs of water stress long before they become visible to the naked eye [16]. Mounted on drones, satellites, or ground-based platforms, these sensors transmit spectral data via IoT systems, allowing farmers to remotely monitor their crops and make informed decisions about irrigation, fertilization, and pest control [17, 18].

Among the most significant applications of these sensors is the calculation of the Normalized Difference Vegetation Index (NDVI), which measures plant health by comparing red and NIR reflectance. Combining IoT with multispectral data enables farmers to optimize water use and improve crop management [19]. For instance,

NIR-based water stress sensors can detect subtle changes in leaf moisture, allowing for precise irrigation management and reducing water wastage in drought-prone areas [20, 16].

In addition to environmental monitoring, plant-based sensors can track key physiological parameters like leaf moisture and chlorophyll content, enabling the early detection of stress factors such as nutrient deficiencies [21]. By integrating multispectral imaging with traditional soil sensors, precision agriculture practices can thrive, providing a holistic view of plant and soil health [22].

This research focused on tackling issues in water stress detection by exploring three key questions:

- **Evaluating the Presence of Spectral Bands with Respect to RGB Images Using YOLO and Faster R-CNN** This research seeks to assess how the inclusion of spectral bands, such as Near-Infrared (NIR) and Red-Edge, improves the detection performance of models like YOLOv8 and Faster R-CNN compared to RGB images alone. The goal is to understand the impact of these spectral bands on the accuracy and effectiveness of object detection models in identifying water-stressed plants.

- **Evaluating the Presence of an Extra Dimension Using YOLO with 4 Channels** This question investigates whether combining multiple spectral bands leads to better detection performance when using YOLOv8 with four-channel inputs. By adding this extra dimension, the study aims to determine if it enhances the model's ability to detect stress compared to traditional RGB inputs.

Chapter 2

Related Works

2.1 Understanding Smart Agriculture

Smart agriculture, also referred to as precision agriculture or digital farming, represents a transformative approach to modern farming. Precision agriculture using IoT, UAVs, big data analytics and machine learning to improve productivity practices and sustainability in farming [23, 11]. These techs made easier the monitoring of environment and crop status in real-time, which promotes an informed decision making that delivers resource use optimization with waste reduction [24]

Smart agriculture has evolved to meet the key global challenges including Food Security, Resource Scarcity and Environmental Sustainability [25, 5]. The global population is projected to reach 9.7 billion by the year 2050 [5] and agriculture systems need to be applicable in order to fulfill increasing demand of the food as well comply with nature resources that are limited such water and arable soils [5]. Therefore, Food Security is one of primary concerns addressed by smart agriculture. With increased global food demands, the agriculture industry will need to be more productive and resilient. The use of smart technologies, e.g., IoT-based systems for precision irrigation and fertilization to ensure crops get the right amount of water or nutrients at optimal times have been shown effective in enhancing crop yields while also reducing environmental footprints [26, 23]. With a multispectral or hyperspectral sensor adapted to UAVs, farmers can obtain images of crop health that allow them to identify problems such as pest infestation or water stress in time to prevent further damage and optimize the use of land and inputs [27, 4].

Climate Change is also a massive threat for agriculture. Changes in weather patterns and more frequent extreme events can cause the variability that farmers rely on to create predictable growing conditions [28]. Also, using big data analytics and machine learning, farmers can anticipate changes in these conditions ahead of time. Combining these technologies with predictive models (built on historical

data and real time monitoring) creates a more resilient farming system which can manage change in climate [29, 24]. For instance, AI models predicting the weather can advise adaptive strategies to ensure crop protection from drought or flooding [30]

Another key in smart agriculture is Environmental Sustainability. Traditional farming practices often lead to soil degradation, water overuse, and greenhouse gas emissions [31]. Smart agriculture tries to decrease these impacts by promoting sustainable practices such as precision irrigation, nutrient management, and carbon footprint monitoring [11, 31]. For instance, technologies that use remote sensing and satellite imagery can monitor soil moisture and plant health, allowing farmers to apply water and fertilizers more precisely, and lead to conserving resources and reducing pollution [32]

2.2 Water Stress Detection

Agriculture is faced by a number of constraints that limit its productivity and sustainability. Some major concerns are identified to be water stress, weed control, diseases detection, and pest management, for which innovative approaches beyond the conventional farming practices are needed [25]. Among the most pressing challenges faced by modern agriculture is the suitable use of water resources. With increasing pressures from climate change and population growth, water availability is becoming more unpredictable, making it essential to manage irrigation practices effectively [33, 34]. Increase in escalating lack of water resources is an important confrontation faced by agriculture [28]. Water stress occurs when the water available to plants is insufficient to meet their physiological needs [34, 35]. This can result from inadequate rainfall, inefficient irrigation practices, or extreme heat, leading to reduced crop growth, delayed development, and lower yields [36]. Water stress can manifest in several ways, from mild to severe. In mild cases, crops may show subtle reductions in growth, while severe stress can lead to wilting, leaf curling, and eventual plant death [37]. Different crops have varying levels of tolerance to water shortages; for instance, drought-tolerant plants can withstand more significant reductions in water availability compared to water-sensitive crops [36]. This often makes crops so affected due to the irregular water availability that reduces yield in crops and, when extreme, leads to total crop failure. Efficient water management is crucial not only for environmental sustainability but also for maintaining productivity, particularly in drought-prone areas [38]. Unfortunately, traditional irrigation methods often lead to over-irrigation or under-irrigation, both of which are costly and hurtful to the health of crops [39]. Therefore, the use of advanced technologies, such as remote sensing, machine learning, and multispectral imagery, has revolutionized how water stress is detected and managed [40, 41].

Through real-time monitoring of crop health, these technologies allow for early detection of stress symptoms, enabling farmers to adjust irrigation schedules and prevent crop losses [42]. Specifically, remote sensing techniques using multispectral bands, such as near-infrared (NIR) and red-edge, can reveal water stress levels that may not be visible to the human eye [40]. These technologies enable proper water management strategies that will conserve water while at the same time allowing appropriate quantities of moisture to reach the plants at the right time. It is in this regard that the successful realization of such solutions will play a vital role in food security, increasing productivity, and reducing environmental marks left by farming operations.

In this study, advanced object detection algorithms, such as YOLOv8 and Faster R-CNN, were applied to multispectral datasets of potato crops to identify regions affected by water stress. The integration of spectral bands such as NIR and red-edge enhanced the model's ability to detect subtle signs of stress, potentially improving the accuracy of water management practices in agriculture. The next section will outline specific methodologies and innovations enabling such advances in modern agriculture.

2.3 Conventional Methods for Water Stress Detection

Traditionally, farmers and researchers have relied on a variety of direct and indirect methods to assess water stress in plants. These conventional methods range from visual observations and physiological measurements to soil-based monitoring techniques. Each method has its advantages and limitations, often influenced by factors such as crop type, environmental conditions, and the scale of agricultural operations.

2.3.1 Visual Observation and Physiological Indicators

Visual observation is one of the earliest and most straightforward techniques for detecting water stress in plants. Farmers have traditionally monitored crops for visible signs of stress, such as wilting, leaf curling, or discoloration. While this method can provide immediate feedback, it is highly subjective and prone to errors. Moreover, by the time visual symptoms appear, the plant may already be significantly stressed, potentially reducing yield [43]. Physiological indicators, such as stomatal conductance and plant water potential, have also been used as reliable measures of water stress [44]. Stomatal conductance, which refers to the rate at which water vapor exits the plant through the stomata, decreases as water stress increases. Measuring stomatal conductance can provide insights into the plant's

water status and help inform irrigation decisions. However, these measurements require specialized equipment and are labor-intensive, making them difficult to scale for large agricultural fields [45].

Another physiological indicator commonly used is the measurement of leaf water potential. This method involves the use of pressure chambers to determine the water content in plant leaves. While leaf water potential provides an accurate assessment of a plant's hydration status, it is invasive and time-consuming, requiring leaf samples to be collected and analyzed under controlled conditions [46]. As with stomatal conductance, this method is not ideal for large-scale monitoring and lacks real-time feedback [47].

2.3.2 Soil Moisture Measurements

Soil moisture is a direct indicator of the water available to plants, making it one of the most common traditional methods for assessing water stress. Soil moisture sensors, such as tensiometers, time-domain reflectometry (TDR) sensors, and capacitance sensors, have been extensively used to monitor water availability in the soil. Tensiometers, for example, measure the tension of water within the soil, indicating the level of water available to the plant's roots. TDR and capacitance sensors, on the other hand, measure the soil's dielectric constant, which changes with moisture content [44].

However, soil moisture sensors only provide localized information and require extensive deployment to cover large agricultural fields, which increases costs and complexity [48]. Additionally, variability in soil features including organic matter content and texture can cause spatial differences in moisture retention across fields, limiting the accuracy of these sensors when used in isolation [49]. Therefore, while soil moisture monitoring remains a critical tool for irrigation management, its effectiveness is often complemented by other sensing methods.

2.3.3 Meteorological Data and Models

Meteorological models have long been used to estimate crop water demand and predict potential water stress. These models rely on inputs like temperature, humidity, solar radiation, and wind speed to calculate evapotranspiration (ET) rates [50]. ET-based irrigation scheduling systems have been implemented in many agricultural areas to guide farmers on how much and when to irrigate based on current weather conditions [39]. Though useful, these models can be inaccurate if they do not account for local variations in soil and plant conditions. Additionally, meteorological approaches provide only indirect estimates of water stress, relying on weather data rather than direct crop measurements.

2.3.4 Remote Sensing Techniques

Remote sensing has emerged as an effective tool for identifying water stress in plants over large areas. Satellite-based and aerial remote sensing platforms can capture vegetation indices like the Normalized Difference Vegetation Index (NDVI) and the Water Index (WI), which provide indirect measurements of plant health [51]. NDVI, for instance, compares the reflectance in the red and near-infrared bands to estimate chlorophyll content and photosynthetic activity, offering an indirect assessment of water stress [52]. Meanwhile, WI is based on the absorption properties of water in the near-infrared spectrum and can directly estimate water content in plant tissues.

Spectral indices offer several advantages over traditional soil and physiological measurements. They provide spatially continuous data across entire fields, overcoming the limitations of point-based soil and physiological measurements . Remote sensing data can be collected frequently and non-destructively, making it ideal for ongoing monitoring of crop health. Though effective in covering large areas, remote sensing techniques are limited by spatial resolution, cloud cover, and the frequency of satellite overpasses [32]. Furthermore, they often require complex data processing and interpretation, which may not be accessible to all farmers.

Multispectral Imaging

Multispectral imaging captures images at particular, distinct wavelengths across the electromagnetic spectrum, commonly focusing on red, green, blue (RGB), near-infrared (NIR), and sometimes red-edge bands. NIR and Red-Edge bands are particularly significant in agricultural applications, as they provide information on the photosynthetic activity and chlorophyll content of plants, both of which are critical indicators of plant health and water stress. For instance, plants that are water-stressed exhibit reduced chlorophyll content, which can be detected in the NIR and red-edge bands as reduced reflectance [6].

By analyzing the reflectance values in these bands, multispectral imaging allows for the computation of vegetation indices like the Normalized Difference Vegetation Index (NDVI), a widely adopted metric for evaluating plant health and detect early signs of water stress. NDVI compares the reflectance values in the red and NIR bands to quantify the amount of chlorophyll present in the plant, which correlates with its overall health and water status. Healthy vegetation typically has high NIR reflectance and low red reflectance, while stressed or unhealthy vegetation exhibits the opposite pattern [49, 19]. This method has been successfully used in a range of crops, from cereal grains to potatoes, providing farmers with early warning signs of stress and allowing for timely interventions.

Hyperspectral Imaging

While multispectral imaging collects data across a restricted set of spectral bands, hyperspectral imaging takes this further by collecting information across hundreds of narrow, contiguous spectral bands. This enables a much more detailed analysis of plant properties, as hyperspectral data provides a continuous spectral signature that can reveal subtle differences in plant physiology that may not be visible in multispectral data [53]. Hyperspectral imaging can detect specific biochemical changes in plants, such as variations in leaf pigments, water content, and cell structure, making it an even more powerful tool for detecting water stress in crops. These changes occur at the molecular level and can be picked up by the hyperspectral sensors before any visual symptoms appear, offering an even earlier detection method than multispectral imaging.

Hyperspectral imaging has been particularly useful in the identification of water stress because it allows for the detection of small changes in the water absorption bands of plant leaves. For instance, the reflectance values in the 970 nm water absorption band are often used to assess leaf water content. A decrease in reflectance at this wavelength indicates a reduction in water content, which is a clear sign of water stress [20]. Moreover, hyperspectral data can be used to generate customized vegetation indices, fine-tuned to detect specific types of stress, including water deficiency, nutrient imbalances, and pest infestations [17].

While the benefits of hyperspectral and multispectral imaging in agriculture are clear, there are still challenges to be addressed. One of the main limitations is the high cost and complexity of hyperspectral sensors, which makes their widespread adoption in agriculture difficult [20]. Additionally, the vast amount of data generated by hyperspectral imaging requires advanced processing techniques, such as machine learning algorithms, to extract meaningful insights from the raw data [16].

However, ongoing advancements in sensor technology and data analytics are likely to overcome these challenges in the near future. As hyperspectral sensors become more affordable and accessible, and as data processing techniques improve, it is expected that hyperspectral imaging will become a standard tool in precision agriculture, enabling even more accurate and efficient monitoring of crop health [17, 22].

Applications in Remote Sensing

The combination of multispectral and hyperspectral imaging with remote sensing technologies, such as satellites, Unmanned Aerial Vehicles (UAVs), and ground-based sensors, has revolutionized the monitoring of agricultural fields. Satellites and UAVs equipped with hyperspectral sensors can capture large-scale data over vast areas, providing a comprehensive view of crop health and water stress levels across entire fields [54]. Remote sensing allows farmers to monitor their crops

remotely and continuously, reducing the need for manual field inspections and enabling more efficient management of resources like water and fertilizer.

One of the key advantages of remote sensing is its ability to monitor crops in real-time. By capturing data over time, remote sensing enables the creation of time-series analyses that can track the progression of water stress in crops. This allows for the identification of trends and patterns that can inform irrigation strategies and other management practices. In addition, remote sensing data can be combined with Geographic Information Systems (GIS) to generate comprehensive maps of crop health, helping farmers pinpoint areas of their fields that require attention [55, 17].

In addition to early detection of water stress, remote sensing can also be used to predict crop yields and optimize resource use. By integrating remote sensing data with other environmental and agronomic data, such as soil moisture and temperature, predictive models can be developed to forecast yield outcomes based on current crop conditions. This allows farmers to enable more informed decision-making regarding when and where to allocate resources, ultimately improving efficiency and reducing waste [18, 49].

2.3.5 Challenges of Conventional Methods

Despite the widespread use of these conventional methods, there are several challenges associated with their implementation. One major limitation is that many of these techniques, such as soil moisture sensors and sap flow measurements, provide point-based data, which may not be representative of the entire field. This spatial variability can lead to inaccurate assessments of water stress if not properly accounted for. Furthermore, many traditional methods require manual intervention or invasive sampling, which can be time-consuming and labor-intensive [6].

Another challenge is the lack of real-time data provided by some methods, such as physiological measurements and satellite-based remote sensing. In precision agriculture, timely decision-making is essential for optimizing irrigation practices and ensuring crop health. Delays in obtaining water stress data can result in sub-optimal irrigation practices, leading to yield losses or unnecessary water use [56].

In conclusion, while conventional methods for detecting water stress have been invaluable in agricultural research and practice, they often fall short in providing real-time, large-scale, and non-invasive monitoring. These limitations have led to the adoption of more advanced technologies, such as multispectral imaging and deep learning, which offer more comprehensive and scalable solutions for managing water stress in modern agriculture [17].

2.4 Deep Learning Techniques for Water Stress Detection

Deep learning (DL) is a subset of machine learning (ML) that has revolutionized the field of agriculture by enabling automated analysis and decision-making. Unlike traditional ML, which relies on manually selected features, DL models automatically learn hierarchical patterns from raw data, making them particularly well-suited for processing complex data types common in agriculture. This includes images from satellites, drones, and sensors, as well as time-series data related to climate, soil moisture, and crop growth patterns.

In agriculture, DL techniques have been successfully applied to tasks such as crop monitoring, disease detection, yield prediction, and water stress detection. One of the primary advantages of DL is its ability to handle large-scale, multidimensional data, such as multispectral and hyperspectral images, which are crucial for detecting subtle variations in plant health and stress levels that are often imperceptible to human observers [11]. DL's capability to integrate various data types into a unified prediction system makes it invaluable in analyzing the complex data common in precision agriculture.

Traditional ML techniques often require extensive preprocessing and feature extraction before the learning process, which can limit their adaptability to different environments and crops. In contrast, DL models are more flexible, able to automatically adjust to new data types and distribution shifts without significant modifications to the architecture. This flexibility is especially beneficial for tasks like water stress detection, where real-time sensor data (e.g., temperature, humidity, and soil moisture) must be integrated and analyzed to make predictions under dynamic environmental conditions [57, 58].

2.4.1 Convolutional Neural Networks (CNNs) for Water Stress Detection

Convolutional Neural Networks (CNNs) have become the most widely used architecture for image-based water stress detection in agriculture due to their ability to automatically extract complex patterns from raw image data. This ability is particularly useful in analyzing data from different sources such as satellite imagery, unmanned aerial vehicles (UAVs), and ground-based sensors, which generate vast amounts of information. CNNs are specifically designed to process spatial data, making them an ideal choice for agricultural applications where spatial relationships, such as the distribution of water stress across a field, are critical.

How CNNs Work in Water Stress Detection

CNNs consist of multiple layers, each performing specific functions such as convolution, pooling, and classification. The convolutional layers automatically learn spatial features from images, such as edges, shapes, and textures, which are critical for identifying subtle signs of water stress in plants. For instance, variations in leaf color and texture, indicative of water stress, can be detected through deep feature extraction. Pooling layers then down-sample the data, reducing its complexity while retaining the most important features. Finally, fully connected layers are used to classify the detected features, determining whether the plant is under stress or healthy. One of the key strengths of CNNs in water stress detection is their ability to handle high-dimensional data such as multispectral and hyperspectral images.

These types of images collect data across a broad spectrum of wavelengths, extending beyond the visible range, such as infrared, which is highly sensitive to plant water content. By processing multispectral images, CNNs can identify signs of water stress long before they become visible to the human eye, providing an early warning system for farmers [59].

The Role of CNNs in Precision Agriculture

CNNs are applied to several types of imagery collected from various sources in agriculture. Satellite-based imagery provides a broad view of large fields and regions, making it suitable for macro-level monitoring. However, satellite imagery may suffer from lower resolution, which limits its ability to detect fine details. UAV-based imagery offers a higher resolution and can be used to capture detailed images of crops at critical growth stages, making it ideal for field-level monitoring [60]. Ground-based sensors, including handheld cameras and stationary devices, provide even more granular data, often capturing the smallest features in plant health and stress levels. In addition to visual data, CNNs can process non-visible spectra, such as near-infrared (NIR) and thermal data, which provide valuable insights into plant physiology. For example, thermal cameras can detect temperature differences in plant canopies that correlate with water stress. When combined with visual and multispectral data, CNN models can provide a comprehensive analysis of plant health, offering real-time detection of water stress with high accuracy [61]

Precision agriculture relies on the ability to detect and respond to small-scale variations in crop health across a field, and CNNs have demonstrated high effectiveness in providing accurate, pixel-level classifications of water stress. These models can segment fields into areas requiring different levels of irrigation or other interventions, helping farmers optimize water use and improve crop yields. For instance, Sankararao et al. [62] applied a CNN model using UAV-based hyperspectral imaging to successfully detect water stress in chickpeas with over 95%

accuracy. This demonstrated the capability of CNNs to identify stressed plants and improve irrigation strategies in real-time. Moreover, Lake et al. [63] demonstrated that CNNs could detect invasive plant species, a sign of environmental stress, using multispectral satellite imagery from Worldview-2 and Planetscope satellites. Their approach highlights how CNNs can process large-scale data to provide actionable insights on plant stress and health.

Additional research by Narvaria et al. [60] applied CNNs to multispectral UAV data to classify different crop types and identify areas under stress. The integration of weather data and crop monitoring allowed for precise decision-making in crop management. This ability to combine different data types makes CNNs particularly valuable in precision agriculture, where multiple variables, such as weather, soil moisture, and plant health, must be considered together. Other works, such as Khaliq et al. [61], explored the refinement of satellite imagery using higher-resolution UAV data for improved stress detection in vineyards. Their CNN-based approach increased the accuracy of stress detection by refining lower-resolution satellite imagery with higher granularity UAV data, demonstrating CNNs' flexibility in managing data from multiple sources and scales. Furthermore, Kamarudin et al. [64] introduced a lightweight CNN model with attention mechanisms to detect water stress in plants, providing a more computationally efficient solution that can be applied even in resource-constrained environments. This type of innovation is critical as precision agriculture becomes more widespread, even in regions with limited access to advanced computational infrastructure. By utilizing CNNs, researchers can improve not only the detection of water stress but also early warning systems that allow for timely interventions. For example, Rojanarungruengporn and Pumrin [65] developed a CNN-LSTM model to detect early-stage water stress in sorghum plants, leveraging phenotyping data. This model could identify stress much earlier than traditional methods, which is essential for preventing yield loss.

2.4.2 Transfer Learning and Pretrained Models for Water Stress Detection

Transfer learning has become integral to agricultural image analysis, particularly for detecting water stress. By leveraging models pretrained on large datasets like ImageNet and fine-tuning them for agriculture, transfer learning enables researchers to bypass the need to build deep learning models from scratch. Commonly used models, such as VGG16 and ResNet, have robust architectures suited for such applications [66]. A comparison of these models is presented in Table 2.1.

Method	Features	Advantages	Challenges	Main Finding	Depth	Dataset	Error Rate	Input Size
AlexNet	Convolutional layers, max pooling, ReLU activation, fully connected layers	High accuracy, efficient for large-scale images, reduced overfitting with dropout	Computationally expensive, large number of parameters	Utilizes Dropout and ReLU	8	ImageNet	16.4	227 × 227 × 3
VGGNet	Deep convolutional layers, small receptive fields, max pooling, fully connected layers	Improved accuracy with deeper layers, simple and uniform architecture	High memory consumption, slow to train due to depth	Increased depth, small filter size	16, 19	ImageNet	7.3	224 × 224 × 3
ResNet	Residual blocks, skip connections, batch normalization	Mitigates vanishing gradient problem, allows training of very deep networks	Complex architecture, increased computational cost	Robust against overfitting due to skip connections	152	ImageNet	3.57	224 × 224 × 3

Table 2.1: Comparison of Deep Learning Architectures for Image Processing

Application of Transfer Learning in Agriculture

Transfer learning is especially valuable in water stress detection due to the limited availability of large labeled agricultural datasets. Collecting labeled data across diverse crop stress conditions can be costly and time-consuming. Transfer learning addresses this by fine-tuning pretrained models, like VGG16 and ResNet, on smaller agricultural datasets. These models, pretrained on ImageNet, have already learned a range of visual features, making them well-suited to detect stress patterns when fine-tuned for agriculture [67].

In practice, transfer learning has shown high accuracy in detecting stressed plants across various crops using hyperspectral and multispectral imagery. For example, Khaliq et al. [61] demonstrated that ResNet-50, when fine-tuned, performed better at detecting water stress in vineyards than models trained from scratch. Similarly, Moiz et al. [68] and Saeed et al. [69] successfully applied transfer learning with VGG16 and InceptionV3 to classify plant diseases and stress in rice and tomato crops, achieving high accuracy.

Advantages of Transfer Learning

Transfer learning significantly reduces the need for extensive labeled datasets. Labeling agricultural images, especially for water stress detection, is labor-intensive and costly. Pretrained models alleviate this by generalizing features from large datasets, allowing fine-tuning with limited agricultural data [70]. Additionally, transfer learning accelerates the training process, making it ideal for projects requiring rapid deployment or with limited computational resources [71].

Challenges and Future Directions

Despite its benefits, transfer learning in agriculture faces challenges, including domain shifts between source (e.g., ImageNet) and target datasets (e.g., agricultural imagery). Pretrained models may not always align with agricultural images,

particularly when dealing with infrared or thermal data. However, fine-tuning and domain-specific augmentation can help address these issues [72].

2.4.3 YOLO in Water Stress Detection

Among the various algorithms for object detection, the You Only Look Once (YOLO) framework remains one of the most notable due to its balance between speed and accuracy. This allows object identification to be done very quickly and reliably within an image. Since the first version of YOLO, several iterations have been released, each improving upon the last to address various challenges. These versions are shown in Figure 2.1. The advantages of YOLO compared to other methods include the following:

- **Speed**: YOLO is extremely fast compared to other object detection methods. The base YOLO model processes images in real-time at 45 frames per second (fps), and a faster version, Fast YOLO, can process images at 155 fps. This makes YOLO suitable for applications requiring real-time processing, such as video streaming and autonomous driving.
- **Simplified Pipeline**: Unlike traditional object detection methods that use complex pipelines involving separate steps for region proposal, feature extraction, and classification, YOLO approaches object detection as a unified regression problem. This involves a single neural network predicting both bounding boxes and class probabilities simultaneously in one evaluation, simplifying the process and making it easier to optimize end-to-end.
- **Global Context**: YOLO considers the entire image during both training and testing, which allows it to reason globally about the image context. This helps in reducing the number of background errors as YOLO can differentiate between objects and background better than methods like Fast R-CNN, which only look at local regions within the image.
- **Generalization**: YOLO has been shown to learn highly generalizable representations of objects. When trained on natural images and evaluated on artwork, YOLO surpasses other methods such as DPM (Deformable Parts Model) and R-CNN (Region-based Convolutional Neural Network). This generalization capability makes YOLO more robust to variations and new domains.
- **Single Unified Model**: YOLO combines all the steps of object detection into a single convolutional neural network, which reduces the complexity of the model and makes it easier to train and deploy. This unified approach

contrasts with methods that require multiple stages and separate training for different components, leading to faster inference and simpler implementation.

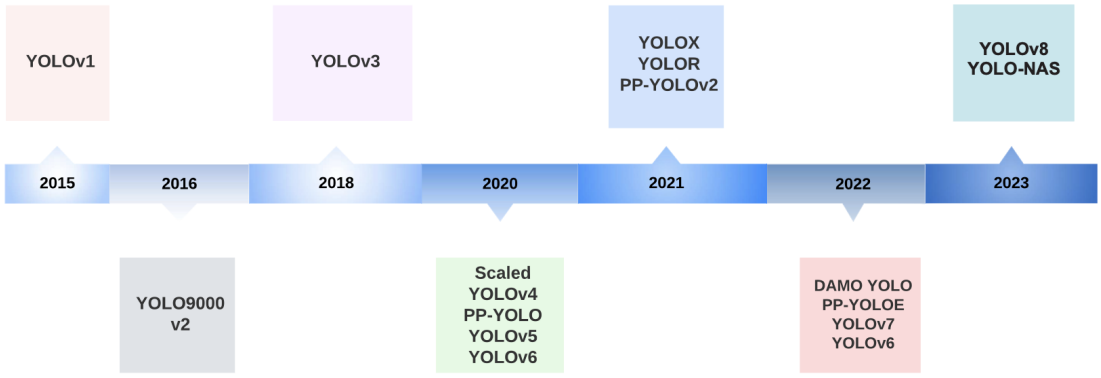


Figure 2.1: Evolution of YOLO versions over time

Unlike traditional object detection approaches, which typically involve two separate steps (region proposal followed by classification), YOLO frames the object detection problem as a single regression problem. This allows YOLO to predict bounding boxes and class probabilities directly from an input image in one pass, hence the name "You Only Look Once. [73].

YOLOv8

YOLOv8 is the latest version of the YOLO object detection family, introduced by Ultralytics. This version continues to build upon the key concepts of YOLO while incorporating several modern innovations to improve both the speed and accuracy of object detection.

2.4.4 YOLOv8 Structure

The YOLOv8 architecture consists of several components that work together to perform object detection. The primary components of the YOLOv8 model include:

Input Layer: The input image is resized to a predefined dimension, often 640×640 or 1280×1280, depending on the model variant (e.g., YOLOv8-small, YOLOv8-medium, or YOLOv8-large).

Backbone: The backbone is responsible for extracting feature maps from the input image. YOLOv8 uses a modified version of the CSPDarknet backbone, which allows for better feature extraction at various scales while maintaining computational efficiency. The backbone is divided into multiple stages, each downsampling

the image and increasing the depth of the feature maps. This hierarchical structure helps capture both low-level features (edges, textures) and high-level features (complex patterns).

Neck (PANet - Path Aggregation Network): The neck component aggregates features from different levels of the backbone. YOLOv8 uses the PANet architecture, which improves information flow across different feature levels. The neck helps fuse high-resolution and low-resolution feature maps to improve object detection for both large and small objects.

Detection Head: The detection head is the part of the network that generates the final predictions. YOLOv8's dynamic head architecture allows the model to predict bounding boxes and class probabilities at multiple scales. Each detection head outputs the following: Bounding Box Coordinates (x, y, width, height) Objectness Score (how likely the bounding box contains an object) and Class Scores (the probability of the object belonging to a specific class).

Figure 2.2 shows the architecture of YOLOv8.

Key Innovations in YOLOv8

Backbone Network Enhancements: YOLOv8 incorporates a more advanced and optimized backbone network, allowing for more efficient feature extraction from the input image. This ensures that even smaller or more complex objects are detected more accurately compared to earlier versions.

Anchor-Free Detection: One significant change in YOLOv8 is the shift towards anchor-free detection. Traditional YOLO models used anchor boxes (predefined box shapes) for detecting objects. However, anchor-based models can introduce complexities, especially when fine-tuning models for specific datasets. YOLOv8 simplifies this by using anchor-free detection, which reduces computational overhead and increases adaptability across different datasets.

Dynamic Head for Object Detection: YOLOv8 introduces a dynamic detection head, which optimizes the bounding box predictions by adjusting the receptive fields dynamically. This results in better localization, especially for objects of varying sizes.

Improved Loss Functions: YOLOv8 uses improved loss functions that weigh the classification, objectness, and localization losses more effectively. This ensures that the model is better at distinguishing between background and foreground objects and provides more precise bounding box predictions.

Post-Processing with NMS (Non-Maximum Suppression): Like previous YOLO versions, YOLOv8 uses non-maximum suppression (NMS) to filter out

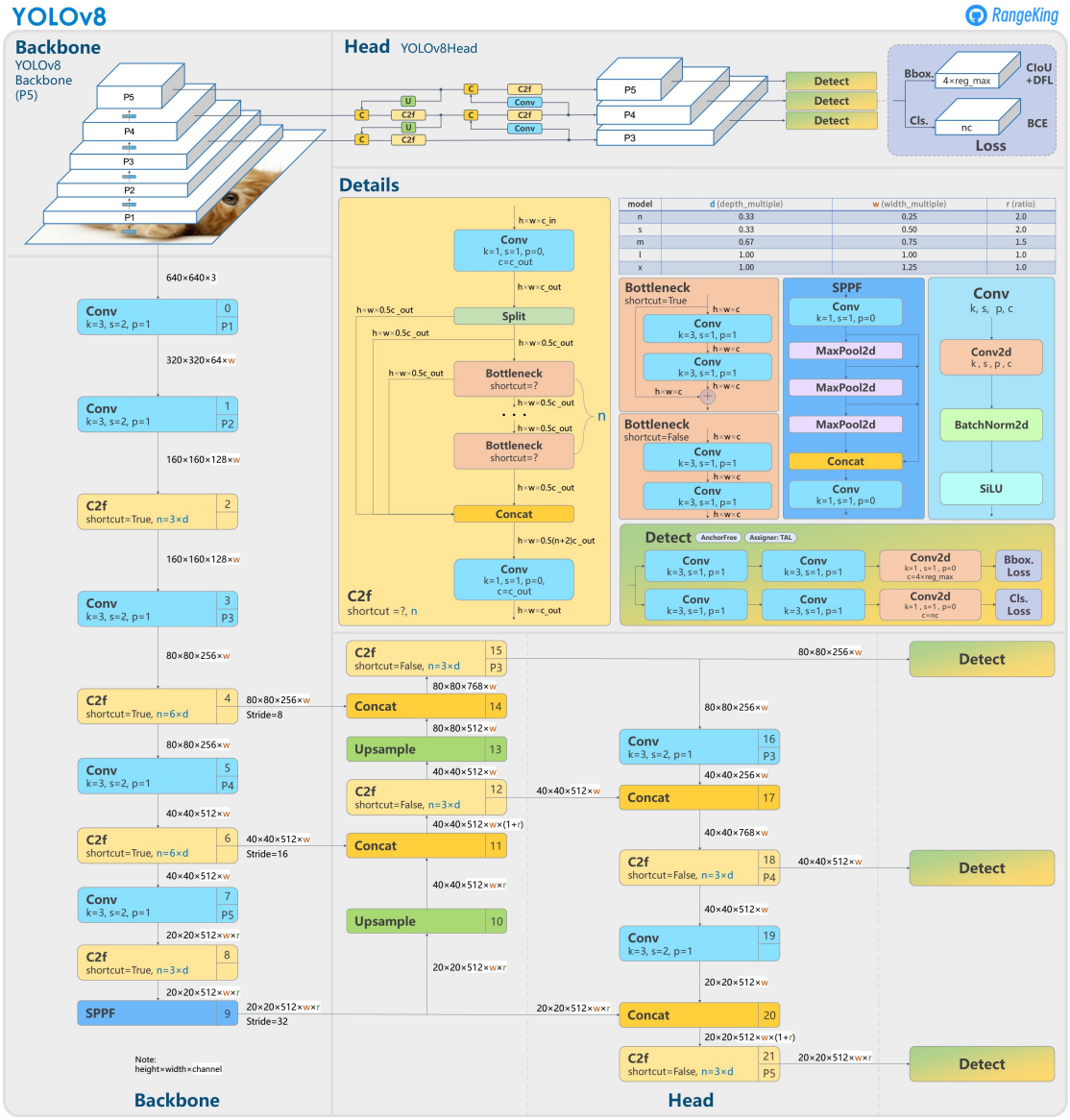


Figure 2.2: Structure of YOLOv8.

overlapping boxes and ensure only the most confident predictions are retained. However, YOLOv8 optimizes the NMS process to further reduce the false positives while maintaining speed.

Multi-Scale Training: To increase robustness, YOLOv8 uses multi-scale training, which allows the model to be trained on images of varying sizes. This improves the model's ability to generalize across different image resolutions during inference.

Application of YOLO in Water Stress Detection

In the agricultural sector, especially in precision agriculture, YOLO-based models have shown significant potential in detecting plant stress, including water stress. Several studies have demonstrated how YOLO and its variants can be applied to multispectral, thermal, and RGB images to monitor crops and detect signs of water stress. Pavani et al. [74] utilized YOLOv8 to develop a plant stress detection system using multispectral imagery. Their model, trained on a combination of visible and non-visible spectra, showed significant promise in identifying stressed plants, especially in detecting temperature anomalies associated with water stress in plant canopies. The study emphasizes the use of UAVs to capture high-resolution imagery of crops, enabling real-time detection of stress at field scale. Guo et al. [75] introduced an improved YOLOv5-based model, DBCR-YOLO, for detecting water surface objects, with potential applications in monitoring water bodies in agricultural fields. Though not directly used for plant water stress detection, the methodology highlights the robustness of YOLO models for detecting small objects in challenging environments, making it relevant for precision agriculture scenarios where detecting early signs of plant stress is critical. Sportelli et al. [76] applied various YOLO models, including YOLOv8, to detect weeds in different turfgrass environments, with implications for broader crop stress detection. Their results demonstrated the superior performance of YOLOv8 in distinguishing between healthy and stressed plants under different environmental conditions, including water stress. This highlights YOLOv8's robustness in real-world agricultural applications. Yue et al. [77] applied an improved YOLOv8-Seg network for the instance segmentation of diseased tomato plants. This model, optimized for segmenting healthy and diseased plants, provides a framework that could be extended to monitor water stress in other crops by analyzing changes in plant morphology. The segmentation of plant regions experiencing stress provides a more granular approach to water stress detection.

Limitations of YOLO

While YOLO models are effective for real-time object detection, they face notable challenges in weed detection. One key issue is their difficulty in detecting small, overlapping objects, a common occurrence in dense vegetation. YOLO's grid-based approach can overlook fine details and precise boundaries, leading to incorrect classifications and missed detections. This is especially problematic in agriculture, where accurately distinguishing between closely spaced weeds and crops is essential. One solution to improve YOLO's ability to detect small, overlapping objects is to increase the resolution of input images. This can help improve the granularity of detection and reduce missed detections caused by the coarse grid of YOLO's architecture. Data augmentation can alson help alleviate this challenge by artificially

expanding the training dataset with a variety of transformations. Techniques like brightness adjustment, flipping, cropping, rotation, and zooming simulate different real-world conditions, making the model more robust to variability.

2.4.5 Faster R-CNN in Water Stress Detection

Faster R-CNN (Regions with Convolutional Neural Networks) is a cutting-edge object detection model that extends the foundational concepts introduced in earlier versions like Fast R-CNN and R-CNN. First proposed by Ren et al. [8] in 2015, Faster R-CNN introduces the Region Proposal Network (RPN), which significantly accelerates object detection tasks by sharing convolutional features with the detection network itself. Unlike earlier methods that relied on external region proposal algorithms like Selective Search, Faster R-CNN generates region proposals in a nearly real-time manner, making it much faster and more efficient [8].

Faster R-CNN Architecture

Backbone Network (Feature Extraction): The backbone of Faster R-CNN is typically a deep convolutional neural network, such as VGG16 or ResNet, which is used to extract feature maps from the input image. These features are crucial for both generating region proposals and for object classification. The convolutional layers of the backbone network extract spatial hierarchies and important features like edges, textures, and object boundaries from the input image. The output is a dense feature map representing the entire image.

Region Proposal Network (RPN): The Region Proposal Network (RPN) is the key innovation in Faster R-CNN. It generates region proposals that likely contain objects by sliding over the feature maps generated by the backbone network. The RPN outputs bounding boxes (regions of interest) and their corresponding objectness scores. For each sliding window, the RPN predicts multiple anchor boxes, which are refined through classification and bounding box regression to select the most promising regions. This step removes the need for external region proposal methods like Selective Search, significantly improving the speed of detection.

RoI Pooling: After generating region proposals, Faster R-CNN uses Region of Interest (RoI) Pooling to extract fixed-size feature maps from the variable-sized regions of interest. This process allows the model to effectively classify objects of varying sizes. RoI Pooling ensures that the feature maps corresponding to each region proposal are fed into the fully connected layers for further classification and bounding box refinement.

Object Classification and Bounding Box Regression: The output from RoI Pooling is passed to a fully connected network for classification and bounding box refinement. Each region proposal is classified into one of the object classes or a "background" class, while the bounding boxes are further refined to tighten the localization around objects.

Figure 2.3 shows the structure of this model.

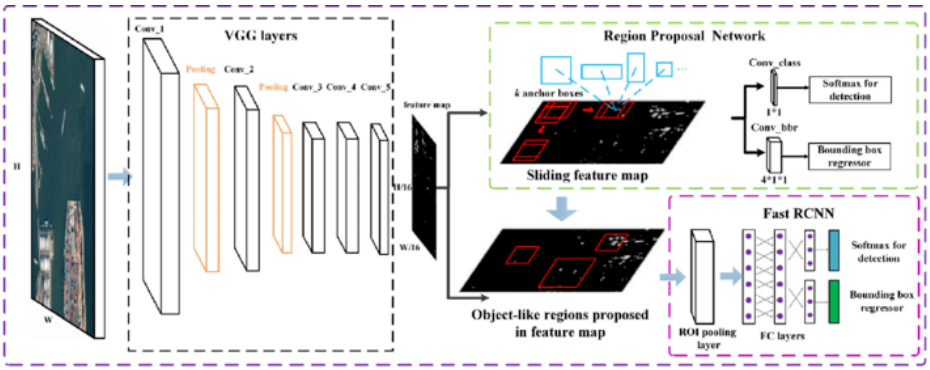


Figure 2.3: Structure of Faster R-CNN.

Key Features of Faster R-CNN

End-to-End Training: One of the biggest advantages of Faster R-CNN over its predecessors (Fast R-CNN and R-CNN) is that the RPN allows the model to be trained end-to-end. This results in faster training and more accurate detections.

Speed: By incorporating the RPN to generate region proposals, Faster R-CNN is significantly faster than Fast R-CNN, which used external region proposal methods like Selective Search. This improvement in speed makes it more suitable for real-time applications, though it's still not as fast as YOLO.

Accuracy: Faster R-CNN achieves state-of-the-art accuracy for object detection tasks due to its two-stage process. The first stage generates region proposals, while the second stage classifies and refines these proposals. This two-step process makes it more accurate than single-shot detection methods like YOLO, especially for complex objects or when high precision is required.

Feature Sharing: Faster R-CNN's architecture shares convolutional layers between the RPN and the final object detection network, making it more computationally efficient than previous methods that processed region proposals and object detection separately.

Applications of Faster R-CNN in Water Stress Detection

While Faster R-CNN is primarily used for object detection tasks, its adaptability to agricultural applications, particularly in water stress detection, has been explored in recent years. Researchers have leveraged Faster R-CNN to detect various stress indicators in crops using a combination of multispectral, hyperspectral, and thermal imagery.

Zhuang et. al. [78] applied neural networks such as Faster R-CNN and YOLO for to evaluate the feasibility of using deep convolutional neural networks for the detection of Florida pusley (Richardia scabra L.) growing in drought stressed and unstressed bahiagrass (Paspalum natatum Flugge). Butte et al. [79] applied deep learning techniques to identify stress in potato crops using aerial imagery. The study applied Faster R-CNN along with other convolutional networks to detect stress in potato crops using aerial imagery. This research provides insights into how water stress-related symptoms, such as leaf discoloration and canopy changes, can be detected with similar methods. The Faster R-CNN model achieved notable accuracy in identifying stressed potato plants, demonstrating its effectiveness in agricultural settings. Zhao et al. [80] applied Faster R-CNN to detect stress in tea plants at the canopy level, specifically addressing three types of stress: tea green leafhopper, anthracnose, and sunburn. This research provides valuable insights into how similar stress symptoms, such as leaf discoloration and deformation, can be detected and segmented using deep learning techniques. The Faster R-CNN model achieved a mean average precision (mAP) of 76.07%, outperforming YOLOv3, which had an mAP of 65.89%, particularly in complex scenarios such as shadow, occlusion, and blurred conditions.

Limitatins of Faster R-CNN

Although Faster R-CNN has shown promissing accuracy in agricultural applications. However, Faster R-CNN's higher computational complexity can limit its use in real-time scenarios, particularly when working with large-scale, high-resolution imagery. To mitigate these limitations, YOLO was integrated for its ability to perform rapid object detection. YOLO's single-stage architecture allows for faster processing, which is essential when UAVs or drones are used to monitor large agricultural fields in real-time.

2.4.6 Key Challenges in Deep Learning for Water Stress Detection

While deep learning has shown promising results in water stress detection, several challenges must be addressed for effective deployment in real-world agricultural applications.

- **Data Scarcity**: One of the key challenges is the scarcity of labeled agricultural datasets, particularly for specific tasks such as water stress detection. Collecting and annotating large amounts of labeled data in agriculture is resource-intensive, as it often involves field visits and expert input. This scarcity of data can limit the performance and generalizability of deep learning models [81].

- **Model Generalization**: Ensuring that models trained in one region or on one crop generalize well to other regions or crops is another significant challenge. Models may face difficulties when applied to different environmental conditions or crop types, especially when there is a domain shift between training and deployment conditions. Data augmentation techniques and transfer learning are often employed to mitigate these challenges and improve generalization [82].

- **Computational Requirements**: Training deep learning models, especially for high-resolution imagery, is computationally expensive. The use of large models such as convolutional neural networks (CNNs) often requires high-performance hardware, which may not be readily available in all agricultural settings. Model compression techniques, such as pruning and quantization, are being explored to reduce the computational load and make the models more suitable for deployment in resource-constrained environments [83].

In response to these challenges, several strategies can be employed to improve the performance of deep learning models in water stress detection.

Transfer Learning: Pretrained models such as VGG16 and ResNet can be fine-tuned on smaller agricultural datasets to reduce the need for large amounts of labeled data. This approach leverages models trained on large-scale datasets like ImageNet and adapts them for agricultural tasks [11].

Data Augmentation: Techniques such as flipping, rotating, and scaling images can artificially increase the size of the training dataset, improving the model's ability to generalize to new conditions. Augmentation is particularly helpful in addressing data scarcity and enhancing model robustness [84].

Domain Adaptation: Domain adaptation techniques are designed to tackle the issue of generalization by adapting models trained on one dataset to perform well on a different, yet related dataset. This is particularly useful when there's a domain shift between datasets (e.g., training on data from one crop or region and applying it to another). One effective domain adaptation method is unsupervised domain adaptation, where models trained on a source domain (e.g., one region or crop type) are adjusted to perform well on a target domain (e.g., another region or crop type) using techniques like adversarial training [72].

To address the aforementioned challenges in this study, pretrained models (VGG16 and ResNet) were used, specifically fine-tuning it on a smaller dataset tailored for water stress detection. Data augmentation techniques were applied to artificially expand the dataset and improve the model's generalization. Additionally, preprocessing steps such as normalization and resizing were carried out to ensure consistency across images, mitigating issues related to variability in the input data.

Chapter 3

Methodology

This chapter details the practical work conducted in this thesis, focusing on the approach and implementations used. The subsequent chapter will present and analyze the results achieved. The chapter begins by describing the initial idea pursued to achieve a functional implementation of Image Segmentation. It explains the decisions made and the reasons behind them. Following this, a description of the final method, Object Detection, is provided, exploring the features and techniques used. Additionally, the configurations employed in each method are discussed in detail.

3.1 Resources

The resources utilized in this project are categorized into three parts based on program complexity and available resources:

- The initial method was conducted on Google Colab, utilizing a T4 GPU/CPU with 12.7 GB of RAM and 107.7 GB of disk space.
- Due to the increased complexity of the second approach (YOLO and Faster R-CNN), resource allocation became more crucial. Initially, it was executed on a Linux server with 31.3 GB of RAM and 8 logical CPUs. Subsequently, due to the requirement for GPU acceleration, it was implemented on a Windows Server equipped with a V100 GPU (32GB vRAM), 22 vCPUs, and 240GB of storage.

3.2 Datasets

For the current research, various datasets have been collected, each tailored to the specific requirements of its corresponding method to achieve optimal results.

3.2.1 Custom Dataset for Image Segmentation

Initially, a segmentation-based approach was considered for assessing plant health, specifically focusing on leaf-level segmentation combined with the Normalized Difference Vegetation Index (NDVI) calculation for each leaf. The rationale behind this method was to enhance model capabilities in monitoring plant health by utilizing near-infrared (NIR) spectroscopy, a widely adopted technique in agricultural research for detecting plant stress. The incorporation of aeroponic cultivation methodologies, as opposed to traditional soil-based methods, further supported the objective of establishing controlled and efficient plant growth environments.

Data Collection

A series of images were captured in a greenhouse outside Turin, Italy. Lettuce samples were used as the primary subject, and a MAPIR S3W RGN camera equipped with a 550, 660, and 850 nm was employed to capture the images. These images were taken under controlled greenhouse conditions to ensure consistency in lighting and plant positioning. The focus of this data collection was to generate a comprehensive dataset that could provide insight into plant health through spectral analysis.

Data Calibration

After the images were collected, calibration of the data was necessary to ensure that accurate and reliable measurements could be derived from the spectral data. The calibration process was conducted using a Calibrated Reflectance Panel, which was placed beside the plants during image acquisition. This panel provided a consistent reflectance surface across both visible and NIR spectral bands, enabling precise compensation for fluctuations in ambient lighting conditions. The reasons for performing this calibration are as follows:

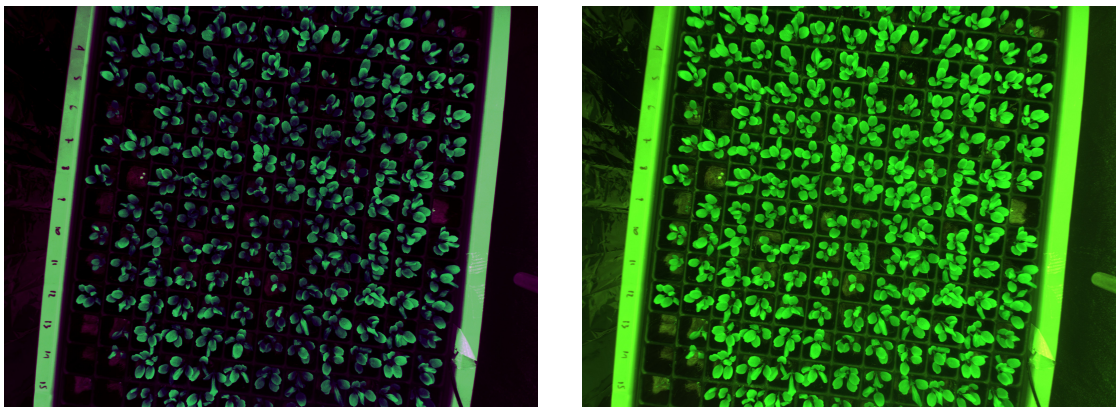
- **Improved Data Quality**: Calibration corrects sensor outputs, eliminating artifacts such as striping and ensuring uniform image quality across detectors and over time.
- **Accurate Temporal Monitoring**: Calibration allows the detection of real environmental changes rather than variations caused by sensor performance, which is critical for longitudinal plant health monitoring.
- **Inter-Sensor Comparability**: When combining data from multiple sensors, calibration ensures accurate comparisons. This capability is vital for applications requiring data integration from different sources, such as environmental monitoring and precision agriculture.

Challenges and Limitations

Despite the potential of this segmentation-based method, several challenges were encountered. The dataset creation process was highly time-consuming due to the logistical challenges associated with collecting images from a greenhouse facility located outside the city. Additionally, the varying stages of plant growth introduced imbalance in the dataset, as plant sizes differed significantly across the captured images. Moreover, while the incorporation of NIR was beneficial for detecting physiological changes, the approach faced practical constraints in terms of image processing and segmentation, particularly given the complexity of real-world plant growth conditions.

Given these challenges, it was determined that the segmentation-based approach may not be the most effective or scalable solution for assessing plant stress. This realization led to the exploration of alternative methods for water stress detection, as described in subsequent sections.

The images collected during this phase of the research are shown in Figure 3.1, and they form the basis of the initial experimental analysis of plant health using spectral methods.



(a) **(b)**

Figure 3.1: (a) Calibrated multispectral image and (b) Raw multispectral image from the generated dataset.

3.2.2 Custom Dataset for Object Detection on Individual Plants

After recognizing the limitations of the segmentation-based approach, a second method was explored, focusing on object detection applied to individual plants. The goal was to create a dataset containing both healthy and stressed plants and then use object detection algorithms to identify and classify the plants in each image. This approach aimed at automating the identification of plant health statuses

without the need for detailed segmentation, thereby streamlining the process.

Data Creation

To capture the necessary data, plants were allowed to grow under different controlled conditions to promote the development of both healthy and stressed plants. The dataset creation process involved capturing high-resolution RGB images.

Data Annotation

In this phase, makesense was used platform for annotation, where bounding boxes were created around each stressed plant to annotate the dataset. The plants were divided into two groups: in the first group, water supply was cut off to induce stress, while the second group continued to receive regular irrigation. Stress in the plants was determined based on the period without water, and visual symptoms such as leaf wilting and discoloration. This annotation allowed the object detection models to distinguish between healthy and stressed plants based on visible signs of stress.

Data Augmentation

In order to enhance the robustness of the dataset and reduce the risk of overfitting, several image augmentation techniques were applied to the collected RGB images. These augmentations included random rotations, horizontal flips, and lighting adjustments to simulate different environmental conditions. The use of augmentation aimed to increase the variability in the dataset and improve the generalization capability of the object detection models.

Data Calibration

Similar to the first approach, a calibrated reflectance panel was used during the image capture process to ensure the consistency and accuracy of the images. This panel was placed beside the plants in each image, providing a consistent reference point for correcting the reflectance values across the visible spectrum. Calibration was essential for mitigating the impact of varying lighting conditions within the greenhouse and ensuring that the captured images accurately reflected the true state of the plants.

Experimenting with Weakly Supervised Learning

During this phase, weakly supervised learning methods were also considered. This approach primarily required counting the number of healthy and unhealthy plants

in each image, rather than detailed pixel-level annotations. Weakly supervised learning allowed us to use less specific annotations while still training models capable of distinguishing between stressed and healthy plants.

For this method, the dataset was annotated by counting the number of healthy and stressed plants visible in each image. Labels were assigned based on the visual inspection of stress symptoms, such as wilting, discoloration, and leaf curling. This approach reduced the need for precise object-level annotations, making it a more scalable solution for large datasets. However, relying on visual cues for stress detection also introduced subjectivity, as human judgment could vary, especially in early-stage plant stress.

Challenges and Limitations

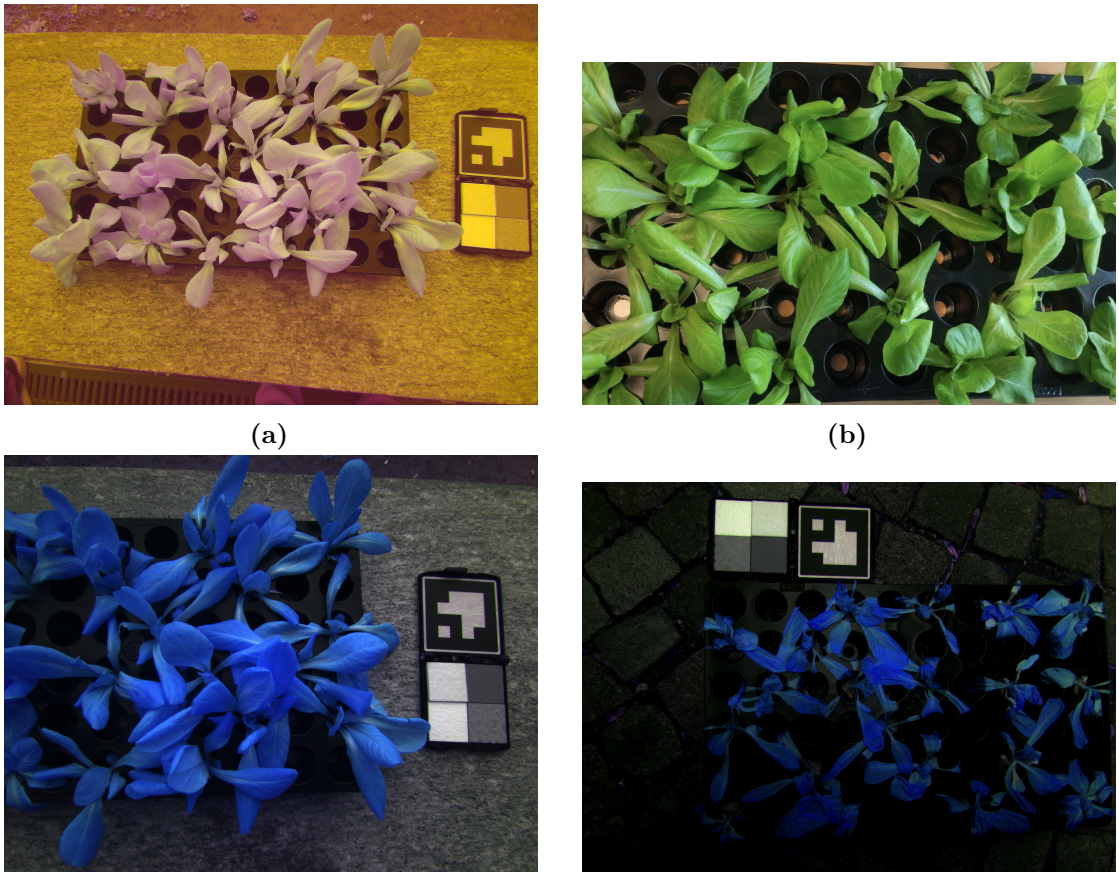
Despite initial progress, this object detection approach faced several challenges. One of the primary limitations was the small number of available plants, which constrained the size and diversity of the dataset. A limited number of healthy and stressed plants made it difficult to achieve a balanced dataset, which is critical for training reliable object detection models.

Additionally, creating stressed or dehydrated plants in a controlled and ethical manner posed significant challenges. Maintaining plant health while simultaneously inducing stress conditions, such as water deprivation, required careful planning, which was difficult to achieve consistently across multiple plants.

Although sensors capable of measuring water stress in plants were considered for labeling the dataset more accurately, the high cost of these devices made this approach unfeasible. As a result, stress labeling relied on visual detection through the naked eye, which introduced further inconsistencies and potential errors into the dataset. Visual stress identification, particularly in the early stages, is subject to human interpretation, making it less reliable for constructing a high-quality dataset.

These challenges made it clear that developing an original dataset for object detection in plant health monitoring would require considerable resources and time. As a result, the focus shifted towards finding existing datasets that better aligned with the objectives of the research. By leveraging publicly available datasets with well-documented stress annotations, the project aimed to continue developing models capable of detecting plant health statuses without the logistical and ethical limitations encountered during the earlier phases.

Figure 3.2 illustrates some of the augmented images generated during this approach.



(a) **(b)**

(c) **(d)**

Figure 3.2: (a) RGN image and (b) RGB image (c) Calibrated RGN image day1, (d) Calibrated RGN image day 4 from the second generated dataset.

3.2.3 Crop Health Assessment Dataset for Object Detection

Aerial images for potato crops acquired at the Aberdeen Research and Extension Center of the University of Idaho was used [79]. The main goal was the acquisition of images for plants under different levels of drought stress through premature senescence. Images were taken using a Parrot Sequoia multispectral camera attached to a 3DR Solo drone. High resolution of 4608 × 3456 pixels for the RGB sensor and four monochrome sensors imaging narrow bands of light wavelengths: green at 550 nm, red at 660 nm, red-edge at 735 nm, and near-infrared at 790 nm—all in the same resolution of 1280 × 960 pixels. The drone was flown at an elevation of about 3 meters for optimal quality images of the potato field. Images of dataset are shown in Figure3.3, Figure 3.4.

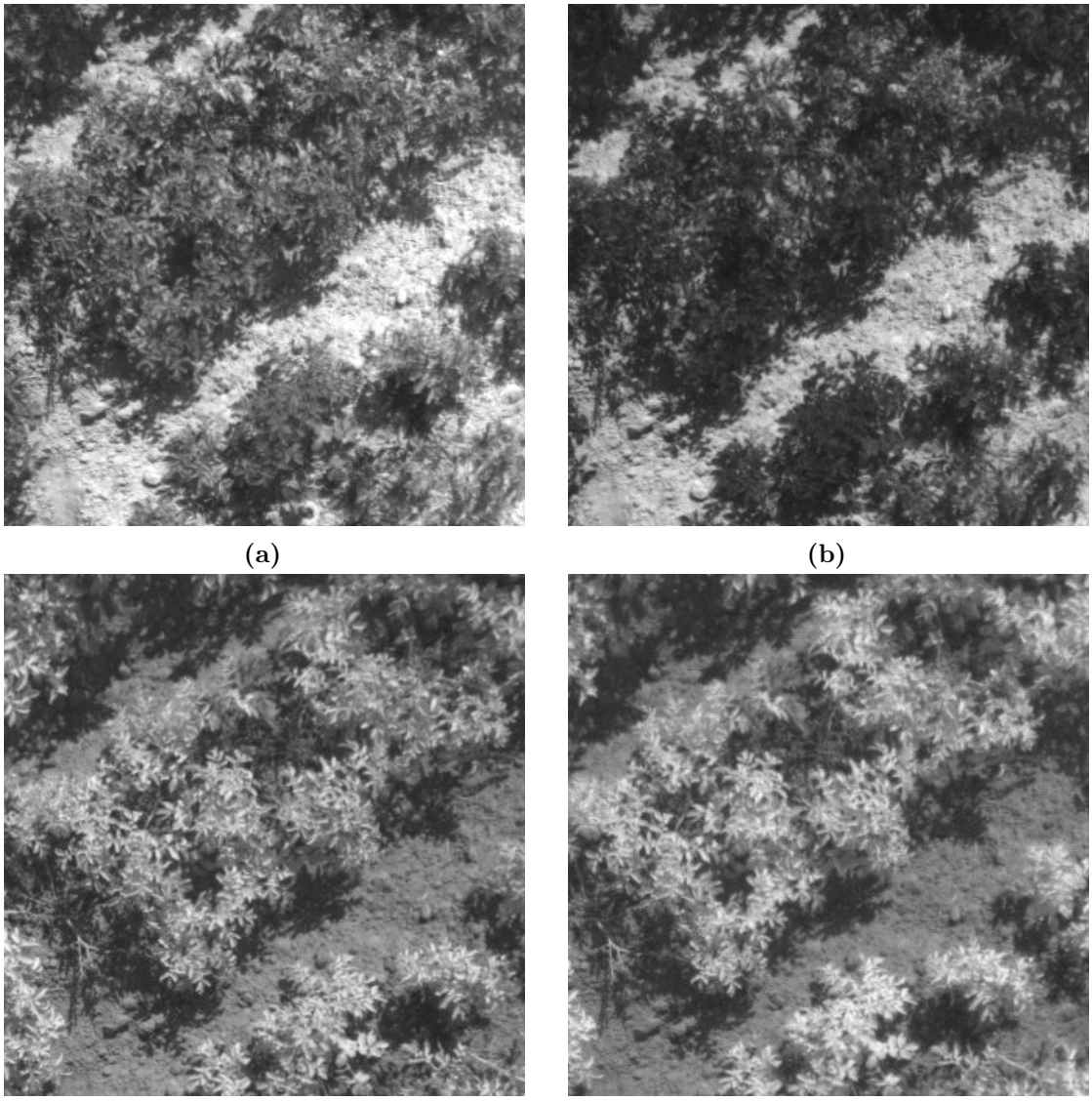
Characteristics of the Dataset

- **Training and Testing**: The dataset is aimed at training machine learning models directed at crop health assessment. It contains high-resolution RGB images and narrow-band multispectral images. Accordingly, a total number of RGB image patches equaled to 360, of which 300 patches were for training, and 60 patches for testing.
- **Annotation Process**: Each of the RGB and multispectral images is related to the ground-truth annotations. The annotations were made in XML and CSV formats and were completed using LabelImg software, which employs rectangular boxes to highlight areas with healthy and stressed plants. Because the difference was only visual, the stressed plants were differentiated as yellow in color, while the healthy plants presented with a more greenish look.
- **Data Augmentation**: Sub-sampling from the full-size RGB images were taken to generate image patches of 750 × 750 pixels after cropping, rotation (in 45, 90, and 135 degree), and resizing operations. The 960 × 1,280 pixels of multispectral images had to pass through processes of undistortion and alignment, with the aim of correcting the sensor positioning, to finally extract patches of 416 × 416 pixels for each spectral band: red, green, red-edge, and near-infrared. Since the number of images is relatively low; hence, it acts as a kind of limitation, and augmentation techniques are applied. These include the scaling of pixel intensities, adjustments with gamma and sigmoid values for brightness and contrast, addition of Gaussian noise, and others. These augmentations were conducted to the 300 training images so as to boost the training set to 1,500 images while the size of the testing set was maintained at 60 images.

The dataset is organized within specific directories:

- **RGB Image Patches**: Data set of 360 images with size 750 × 750 pixels, 300 for training, and 60 for testing.
- **RGB Image Patches - Augmented Dataset**: It is having 1500 augmented training images and 60 testing images.
- **Spectral Image Patches**: It consists of 360 images, 416×416 pixels each, and has been created separately for the four spectral bands. Out of these 360 images, there are 300 training images.
- **Spectral Image Patches - Augmented Dataset** : This dataset comprised of 1500 spectral image patches to be used as a training dataset and 60 others for testing for each number of bands.

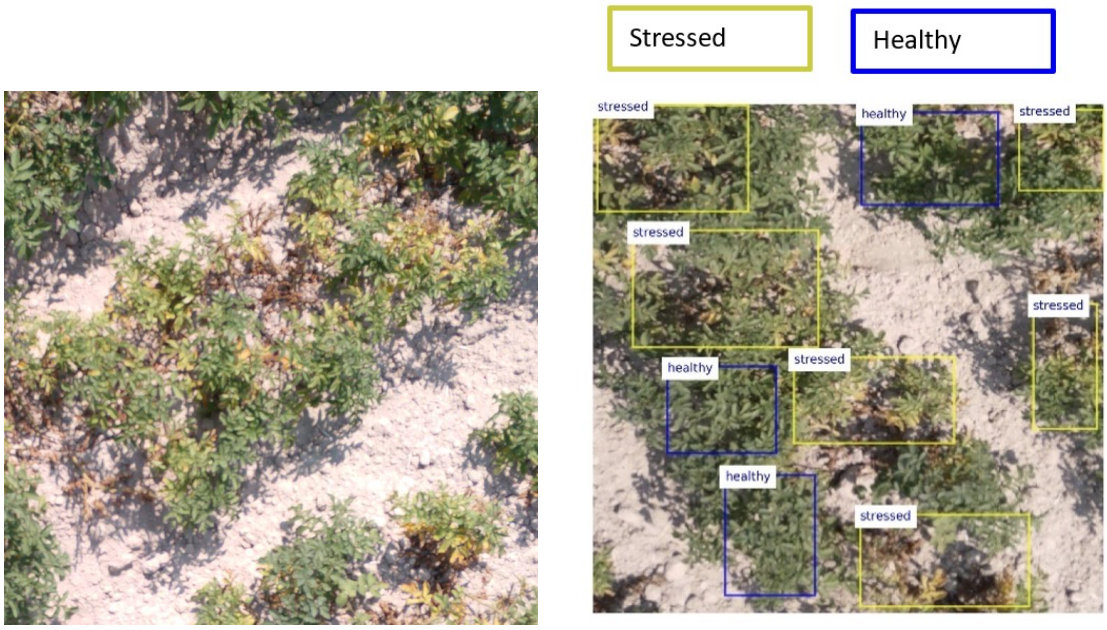
This massive dataset has been publicly available, and it is going to be very important in object detection model development and validation with crop health monitoring studies. The dataset is available at the University of Idaho's website.



(a) **(b)**

(c) **(d)**

Figure 3.3: Spectral images without augmentation: (a) Green channel (b) Red channel (c) Red-Edge channel and (d) is Near-Infrared channel image from the Potato Crop Health datasset.



(a) **(b)**

Figure 3.4: (a) RGB image without augmentation (b) Example of how annotation is applied to images.

3.3 Initial Approaches: Image Segmentation and Vegetation Indices

As discussed in earlier chapters, one widely used method in agricultural studies for plants is Image segmentation which can be classified into three main approaches: color-based, threshold-based, and learning-based methods. For this research, color-based and threshold-based segmentation methods were selected for implementation due to their relative simplicity and effectiveness.

The segmentation method was chosen for two main reasons:

- It is straightforward, particularly in terms of complexity, for both color-based and threshold-based approaches.
- It offers acceptable accuracy and effectiveness, especially in real-time applications.

Additionally, this method can be integrated with vegetation indices such as NDVI (Normalized Difference Vegetation Index). As explained previously, NDVI is utilized to assess plant health and vegetation density due to:

- NDVI is widely used to assesse plant health and vegetation density due to its widespread adoption simplicity in calculation.

- Combining image segmentation with NDVI provides valuable insights with calculating the NDVI index for each segmented plant or leaf and analyzing its health within its range.

To calculate this index, near-infrared and red channels are considered. Healthy plants reflect more NIR (near-infrared) and absorb more red light due to their chlorophyll content. The formula to calculate NDVI is:

$$\frac{NIR - Red}{NIR + Red}$$

NIR: Plants reflect NIR light, and healthy vegetation reflects more of it compared to unhealthy or stressed vegetation.

Red Light: Chlorophyll in plants absorbs most of the red light, so a healthy plant will have a lower red light reflectance.
The NDVI value ranges between -1 and +1:
Values close to +1 indicate healthy, dense vegetation.
Values near 0 suggest bare soil or unhealthy vegetation.
Negative values typically indicate water, snow, or clouds.

3.3.1 Tools and Libraries Used

The libraries employed in this step include:

- **OpenCV** an open-source software library for computer vision and machine learning, is extensively used for image processing applications. It offers a variety of functions for tasks such as object detection, face recognition, and image segmentation. In the proposed method, OpenCV is used for its efficient image processing capabilities to manipulate and analyze visual data [85].
- **NumPy** is a core library for numerical computations in Python. It provides support for large multi-dimensional arrays and matrices, along with an extensive set of mathematical functions to manipulate these arrays. This library is crucial for efficiently managing and processing numerical data in the proposed system [86].
- **Matplotlib** is a versatile library for generating static, animated, and interactive visualizations in Python. The pyplot module is particularly useful for generating plots and graphs to visualize data. In the proposed method, Matplotlib is used to plot and display images and results, facilitating the interpretation of the processed data [87].
- **Scikit-image** is an image processing library that builds on the capabilities of NumPy and SciPy. It provides a range of algorithms for image segmentation, geometric transformations, analysis, and filtering. Functions such as clear_border,

measure, label, and regionprops are utilized in the proposed method to segment and analyze images effectively [88].

- **SciPy**, particularly the ndimage module, offers various functions for multi-dimensional image processing. This includes operations like measurements, center of mass calculation, binary dilation, and zooming of images. The proposed method leverages these functions to perform complex image processing tasks that require precise manipulation of image data [89].
- **Plotly** is an interactive graphing library that enables the creation of highly customizable and interactive visualizations. The graph_objects module provides a low-level interface for building figures, while plotly.express offers a higher-level interface for quickly creating common visualizations. In the proposed method, Plotly is used to create interactive plots and visualizations to better understand and present the processed image data [90].

Below the algorithm for the proposed method is provided:

Algorithm 1 Plant Segmentation and NDVI Calculation

Data: Input image
Result: Segmented regions with NDVI calculation
Input : Image
Output : NDVI for segmented regions

`// Reading the picture`
Read the input image

`// Preprocessing`
Detect vertical lines of the container around the plants
Rotate the picture by 90 degrees
Remove the frame

`// Segmentation: combining color-based and threshold approaches`
Find minimum and maximum pixel values in the picture
Determine the threshold for plant detection using Otsu's method or manual examination
Create a mask and fill holes to obtain the plant shapes

`// Calculating NDVI`
foreach *region in segmented image* **do**
  Calculate NDVI for each region
end

To refine the methodology, the NDVI for each leaf and then eaxh plant was calculated to provide insights into the health status of individual leaves. These individual index values were then compared with the NDVI calculated for the entire image to assess the overall plant health.

3.4 Final Approach: Object Detection with Deep Learning Methods

3.4.1 Preprocessing Steps

Preprocessing is a fundamental phase in the preparation of data for machine learning tasks, particularly in the field of object detection. It involves a series of essential steps aimed at refining raw data to meet the specific requirements of models such as YOLO. The steps sompleted for this stage include tasks such as standardizing data formats, organizing dataset path to match with YOLO configuration, and training and validation splits. It is not only about formatting data; it is about ensuring everything is suitable so that models can learn accurately and perform well. For Faster R-CNN, the annotation format of the dataset matched the accepted annotation of the model, therefore there was no need for altering the annotation given by the dataset.

Figure 3.5: Label formats in YOLOv8

Tools and Libraries Used

For this step, in addition to Numpy, the following libraries are also used:

- **Os** is a Python module that provides functionalities specific to the operating system, such as reading and writing files within the file system [91].

- **Shutil** is a module offering high-level file operations, including tasks like copying and removing files or directories [92].
- **Random** is a Python library used to generate pseudo-random numbers for various data distributions, including both integers and floating-point values [93]
- **Pandas** is a powerful Python library for data manipulation and analysis, offering data structures like DataFrames for efficient handling of structured data [94].
- **PIL (Pillow) image** is a widely-used library that supports a variety of image file formats and provides efficient image processing features[95].
- **Tifffile** is a Python library for reading and writing TIFF images, supporting multi-page and high-bit-depth files [96].
- **Pathlib** is a Python module designed to work with filesystem paths in an object-oriented way, making path operations more intuitive and easier to use [97].
- **Collections** is a module that provides specialized data structures like named-tuples, deques, and defaultdicts, as alternatives to Python's built-in containers [98].
- **Yaml** is a library that allows for the serialization and deserialization of Python objects into YAML format [99].
- **Sklearn.model_selection** is a module within scikit-learn that includes utilities for model selection, validation, and evaluation, such as train/test splitting and cross-validation [100].
- **Glob** is a module that searches for all file paths matching a specific pattern, based on Unix shell rules, and returns the results in any order [101].

Preprocessing Functions

- **Modifying annotation format**: The labeling format used in YOLOv8 follows the {class, x_center, y_center, width, height} format, requiring box coordinates to be normalized within the range of 0 to 1 (see Figure3.5) [102]. Consequently, the initial preprocessing step involves converting dataset labels, which are stored in rows of CSV files for each object in an image (and also in XML files for each image) in {filename, xmin, ymin, xmax, ymax, class} format, to match YOLOv8's requirements. This conversion is achieved through a function named **convert_df_to_yolo_format**.

- **Label corrections**: It is necessary to ensure that the names of labels and images match; therefore, the necessary adjustments are made using the **rename_labels** function.

- **Organizing Dataset Path** The dataset path must follow this structure for YOLOv8: 'data/images/train' and 'data/labels/train' (similarly for validation and test folders). This format is essential for the YAML configuration files

used by YOLO, which specify the dataset's root directory and the relative paths to the training, validation, and testing sets. In order to match with this format, **move_matching_labels** function is used.

- **Fix Train/Validation Split**: Splitting data into training, validation and test sets is an important step in the development of machine learning models. Several benefits in this context include:
  - Hyperparameter Optimization: Tuning the hyperparameters is possible with the validation set. In other words, optimization can be performed on the model by this set for the hyperparameters to ensure high performance against unseen data [103].
  - Early Stopping: You could monitor the performance on a validation set and then stop training when its scores are weak. This will prevent overfitting in the sense that training would be stopped when the performance started deteriorating on the validation set [104].

  For this project, Two types of splits were organized for the train, validation folders. A fixed train/validation split: the training folder of the dataset was split at a 0.2 ratio for the validation stage. This splitting was performed with **train_val_test_split**.

- **K-Fold Cross-Validation**: k-fold cross-validation is a powerful technique in machine learning for evaluating the performance and robustness of a model. The following reasons support the importance of this method in this project application:
  - **Maximizes Data Usage:** In k-fold cross-validation, data are divided into k subsets, and the model is trained and validated k times, each time with a different subset as the validation set and the remaining k-1 subsets as the training set. This way, all data points are used both in the process of training and validation, maximizing the use of available data [103].
  - **Augments Robust:** A k-fold cross-validation is conducted in order to validate the model with k different subsets of the data, and it will help to give a more robust estimate of the model in detecting and avoiding overfitting, as this means that the model will perform well across all the folds [105].
  - **Provides Reliable Performance Estimates:** The average over all k-folds will be considered the final performance metric. This actual step of averaging, in fact, reduces variance of the estimate of performance and gives a more reliable measure of the true performance of the model

with unseen data [106].This research implemented the method using resources from both Ultralytics documentation, and it is contained in the k_fold_cross_validation.ipynb file.

- **Creating Multispectral Images**: As elaborated, the crop health dataset possesses spectral bands such as near-infrared, red edge, red, and green. One of the research objectives considers the impact of multispectral channel images in the detection of plant health status. Therefore, the **combine_channels** function merges these bands into "RGREN" (red, green, red edge, near-infrared), "RGN" (red, green, near-infrared), and "RGE" (red, green, red edge) imagery. The combine channels function receives a combination type, the four bands, and an output folder as input. This function converts each channel into a numpy array and uses numpy's stack function to merge these bands. Some of generated images for each combination type is provided in Figure3.7

Data Augmentation

The models were explored using both non-augmented and augmented datasets. However, to increase the dataset size and improve the model's robustness, the decision was made to move forward with augmented data. Various augmentation techniques were applied to enhance the diversity of the training samples. The following augmentations were used:

- **Blur**: A slight blur effect was applied with a probability of 0.01, using a blur limit between 3 and 7. This helped simulate image distortions that might occur in real-world scenarios.
- **Median Blur**: Similar to blur, this technique applied a median blur with a probability of 0.01, within the same blur limits (3, 7), ensuring more variation in image sharpness.
- **ToGray**: A grayscale transformation was used with a probability of 0.01, keeping the number of output channels at 3, using the 'weighted_average' method to simulate grayscale imagery and challenge the model's color-based detection capabilities.
- **CLAHE (Contrast Limited Adaptive Histogram Equalization)**: With a probability of 0.01, CLAHE was used to enhance local contrast, with clip limits ranging from 1 to 4.0 and a tile grid size of (8, 8). This improved the model's ability to detect details in low-contrast regions.
- **Mosaic Augmentation**: This method took 4 images and combined them into a single image. Mosaic augmentation resized each of the four images,

stitched them together, and then took a random cutout from the stitched images to create the final mosaic image. This technique was particularly useful for providing diverse and complex training examples to the model.

3.4.2 Hyperparameters

The YOLO model was tested using various hyperparameters to evaluate its performance under different conditions. Learning rates were adjusted and tested at value of 0.1 for AdamW optimizer. Additionally, experiments were conducted with varying batch sizes, including 8, 16, and image sizes of 640 and 416 pixels. Different pretrained models were also explored, including YOLOv8n, YOLOv8s, YOLOv8m, to assess their effectiveness. The YOLO model was also tested with different numbers of epochs, including 50, 80, and 100, to observe how training duration affected performance. Additionally, it was noted that the default data split provided suboptimal results due to an uneven data distribution. To address this, the dataset was shuffled, and a custom split was implemented, dividing the data into 80% for training, 10% for validation, and 10% for testing. This reshuffling ensured a more balanced distribution, improving the model's ability to generalize across all phases of training and evaluation.

YOLO Hyperparameters	**Values**
Epochs	80, 100
Batch Size	8, 16
Loss Function	CIoU loss, DFL loss, Varifocal Loss
Activation	Mish
Optimizer	AdamW
Evaluation Metric	Precision, Recall, MAP50, MAP50-95

Table 3.1: YOLOv8 Hyperparameters and Their Values

Faster R-CNN was evaluated using RGN, RGE images. Due to the significant time and resource demands, the model was tested with a batch size of 16, using SGD with a learning rate of 0.01 over 100 and 50 epochs.

Faster R-CNN Hyperparameters	**Values**
Epochs	50,100
Batch Size	16
Loss Function	classification and regression loss
Activation	ReLU
Optimizer	SGD
Evaluation Metric	Precision, Recall

Table 3.2: Faster R-CNN Hyperparameters and Their Values

3.4.3 Proposed inputs

To start the implementation, the following dependencies were required to install:

Listing 3.1: Required dependencies

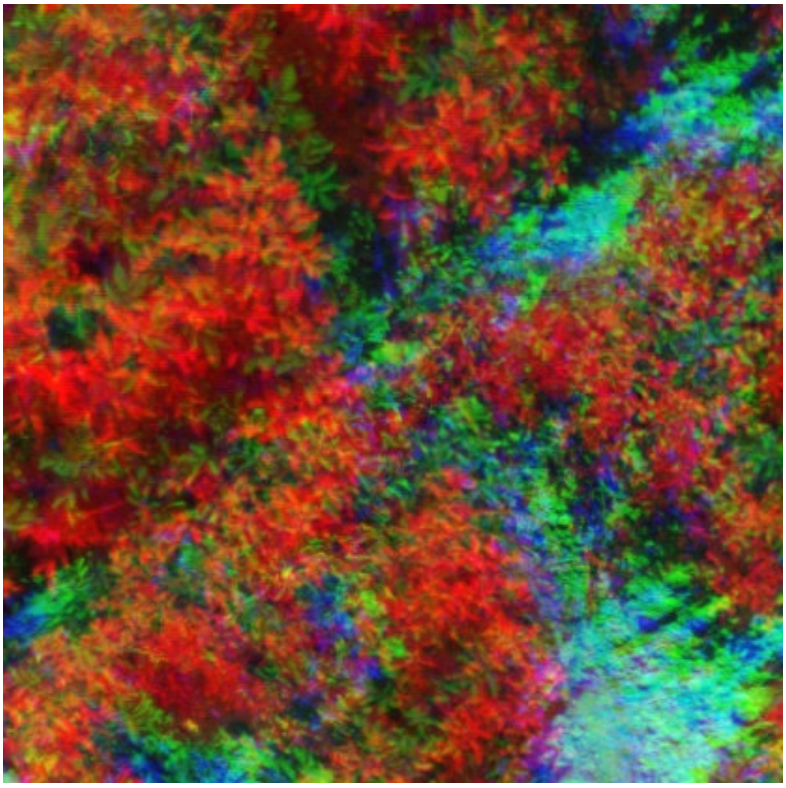
```
dependencies = [
    "numpy>=1.23.0,<2.0.0",  # Temporary patch for compatibility
    errors
    "matplotlib>=3.3.0",
    "opencv-python>=4.6.0",
    "pillow>=7.1.2",
    "pyyaml>=5.3.1",
    "requests>=2.23.0",
    "scipy>=1.4.1",
    "torch>=1.8.0",
    "torchvision>=0.9.0",
    "tqdm>=4.64.0",  # Progress bars
    "psutil",  # System utilization
    "py-cpuinfo",  # Display CPU info
    "pandas>=1.1.4",
    "seaborn>=0.11.0",  # Plotting
    "ultralytics-thop>=2.0.0",  # FLOPs computation
]
```

In this work, various input combinations were explored using YOLOv8 for the detection of water stress in potato crops. The input channels included RGB, RGN (Red, Green, Near-Infrared), RGE (Red, Green, Edge), RGREN (Red, Green, Edge, Near-Infrared - a 4-channel input) and RGBN (RGB + NDVI). For the implementation of Faster R-CNN, the RGN (Red, Green, Near-Infrared) combination was used.

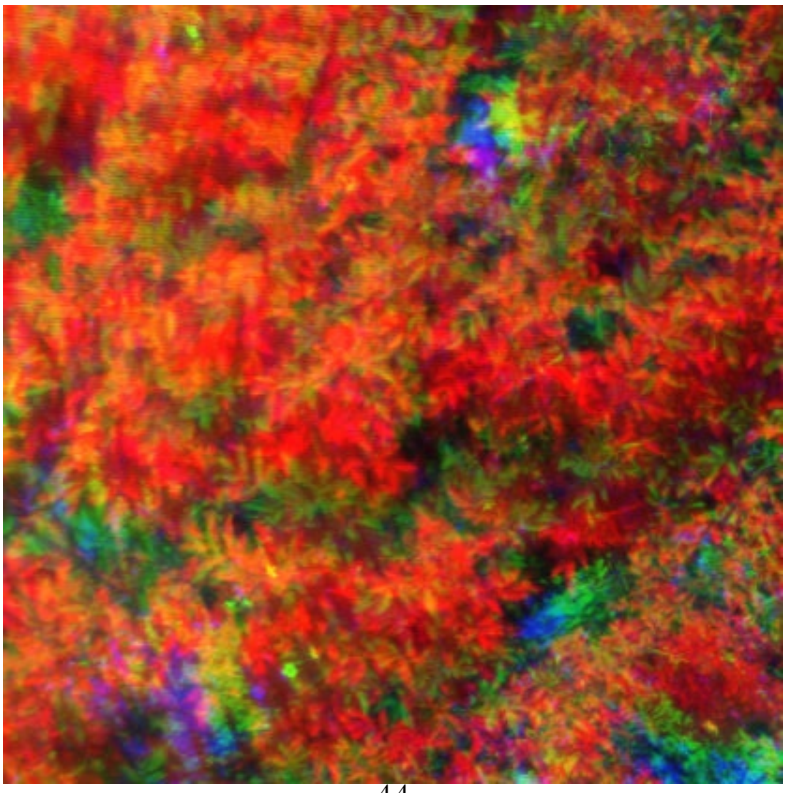
For the 4-channel configuration RGREN and RGBN, a combination of modifications were applied which will be discussed in the next section. These adjustments allowed the model to utilize the detailed color information from the Red and Green channels, combined with the spectral richness provided by the Near-Infrared and Edge channels. The combination of these channels provided a broad spectrum of information, particularly useful for identifying signs of water stress, as Near-Infrared (NIR) can capture details about plant health not visible in the RGB spectrum.

The 3-channel configurations included RGN and RGE, where RGN provided a significant advantage by incorporating the NIR channel, which helps in distinguishing stressed crops from healthy ones. The NIR channel captures light reflection from the plants, which typically reflects more NIR light when healthy. This enhances the ability to monitor the health status of the crops. The RGE configuration (Red, Green, Edge) was particularly useful for focusing on structural differences in plant morphology, where the edge information highlighted boundaries between plants, aiding in more precise object detection. Each combination was tailored

to leverage different spectral properties, with the NIR and Red Edge channels being particularly effective in identifying water stress due to their sensitivity to plant health. These different input configurations were essential in evaluating how different spectral bands contribute to the task of water stress detection in potato crops, optimizing the YOLOv8 and Faster R-CNN models for this purpose. Following figures show examples of the different channel combinations used.

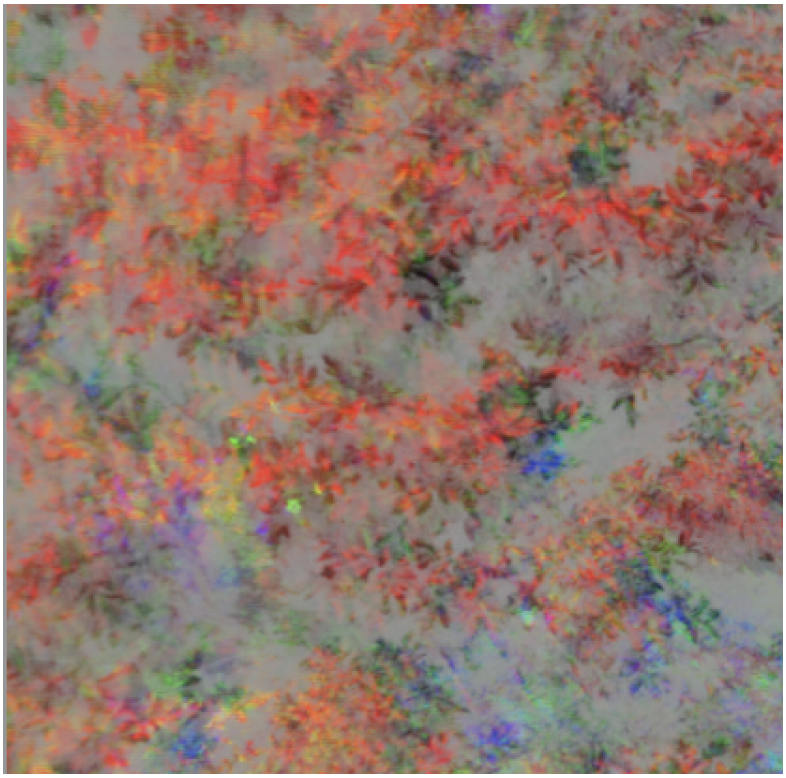


(a)



(b)

Figure 3.6: Spectral images: (a) Red, Green and Near Infrared combination (b) Red, Green and Red Edge combination.



(a)

(b)

Figure 3.7: Spectral images: (a) Red, Green, Red edge and Near Infrared combination (b) RGB+NDVI (RGBN).

3.4.4 Model Architecture Modifications

To modify YOLOv8 to support 4-channel input, several critical adjustments were made across various components of the model to ensure proper handling of multispectral images and the new input structure. The model's configuration files located in the 'models/v8/' folder were modified to support 4-channel input by adjusting the 'channels' attribute. This change ensures that the model architecture itself can process images with an additional channel, such as those in multispectral imaging. YOLOv8 was originally designed to handle 3-channel (RGB) images, so updating the channels attribute was a necessary first step. To handle 4-channel data efficiently, modifications were made to the image loading pipeline in files such as 'converter.py', 'loaders.py', 'utils.py', 'split_dota.py', and 'dataset.py'. Specifically, OpenCV's 'IMREAD_UNCHANGED' function was used to load images with all channels intact. This method allows for the seamless reading of images that contain an additional channel beyond the standard RGB setup. These changes were crucial in ensuring that the data preprocessing pipeline is aware of the extra channel and processes it correctly throughout the data loading phase. The next set of changes involved updating the data augmentation pipeline to accommodate 4-channel images. Files such as 'augment.py' and 'base.py' were modified to ensure that any transformations applied to the images during data augmentation were also correctly handling the 4-channel input. This was necessary because standard augmentation techniques (like flipping, rotating, or color jittering) need to be adjusted to work with images that contain an additional channel. The data augmentation transformations were rewritten to preserve the integrity of the extra channel during these processes. In the engine section, including files like 'validator.py', 'auto_backend.py', and 'tasks.py', necessary changes were made to manage the model's 4-channel input during training, particularly in the model's warm-up phase and batch processing. Specifically, the batch normalization layers and other internal processes were updated to accept 4-channel input shapes, meaning the model can now handle input dimensions such as '(batch_size, 4, height, width)'. This step ensured that the neural network components would correctly handle the increased dimensionality during both forward and backward passes. The model's initialization process, including its warm-up phase, was explicitly configured to manage the extra channel, preventing potential input shape mismatch errors during training. Another important aspect involved modifying the mosaic data augmentation functionality in the 'utils/plotting.py' file. YOLOv8's mosaic augmentation, which combines multiple images into a single composite image for training, was updated to properly integrate the additional channel. This ensures that during training, the model is not only combining the RGB channels of images but also preserving and utilizing the extra multispectral channel. Proper handling of this step was crucial for data augmentation techniques, allowing the model to

effectively learn from multispectral data while benefiting from the regularization that mosaic augmentation provides. The dataset YAML configuration file was updated to reflect the new 4-channel input structure. This change ensured that the model reads the correct input format during both training and evaluation. The YAML file defines how the model should interpret and preprocess the data, and updating this file for 4-channel input ensured that the data pipeline, from loading to training, was consistent with the new input format. For multispectral image types like RGN, RGE, and RGREN, the backbone layers of the YOLOv8 model were frozen during training. This strategy was implemented to allow the model to focus on learning the parameters specific to the new data type without altering the learned features in the backbone network. By freezing these layers, the model could focus on fine-tuning the head of the network for feature detection in multispectral images, improving performance for specific tasks like water stress detection in crops.

As part of the modification process for fine-tuning the Faster R-CNN model for a custom multi-object, multi-class detection task, key adjustments were made to specific layers within the model. Using PyTorch's implementation of Faster R-CNN, pre-trained on the COCO dataset, the model was adapted to detect multiple classes, specifically focusing on healthy and stressed plants across several spectral combinations: RGN and RGE. Modifications to Faster R-CNN are including: Box Predictor (Classifier) Modification: The original box predictor, responsible for classifying detected objects, was designed for the COCO dataset, which includes 80 classes. This predictor was replaced with a new classifier tailored to the custom dataset. The pre-trained model was loaded using 'torchvision.models.detection.fasterrcnn_resnet50_fpn'(pretrained=True), based on the ResNet50 backbone with a Feature Pyramid Network (FPN). First, the number of input features for the existing classifier was extracted, and then the 'box_predictor' layer was replaced with a new 'FastRCNNPredictor', which uses the extracted input features but outputs predictions for the new dataset. The number of classes was set to 3, representing 2 foreground classes (healthy and stressed plants) and 1 background class. Handling the Custom Dataset: Since the model was designed for multi-object, multi-class detection, a custom dataset was required. One of the primary challenges faced was handling the diverse object classes and bounding box annotations within the custom dataset. In PyTorch, the dataset was structured to return both the image and corresponding annotations in a format suitable for Faster R-CNN, which included bounding box coordinates and labels for each object in the image. This made dataset preparation a crucial and complex step in the pipeline.

A key limitation encountered was the high computational time required to fine-tune this model. The complexity of the model, combined with the need to process multi-channel images, significantly increased the time required for training.

Due to limited computational resources, it was not feasible to scale up experiments as extensively as initially planned, which restricted further optimization and testing with larger datasets. This process demonstrates the trade-off between model accuracy and computational efficiency, emphasizing the importance of balancing model complexity with available resources.

3.4.5 Further Experiment with Pretrained Model

To further explore the capabilities of deep learning in detecting water stress, the YOLOv8s Leaf Detection and Classification model was employed as part of the experimentation. This model, originally designed for the classification of various leaf types, offers robust real-time object detection and classification features. By utilizing this pretrained model, the aim is to evaluate its performance in identifying stress levels in potato crops under water-limited conditions.

The YOLOv8s Leaf Detection and Classification model is constructed on the YOLOv8 architecture, known for its efficiency and accuracy in object detection tasks. Pretrained on a dataset with a wide variety of leaf classes, this model is capable of detecting and classifying multiple leaf instances within an image, assigning them to specific categories based on visual features. The model is trained to recognize a variety of plant types, including food crops, fruit-bearing plants, vegetables, and other significant species. Notably, it is well-suited to identify leaves from plants such as potato, corn, wheat, soybean, and tomato, all of which are relevant to agricultural studies. The YOLOv8s Leaf Detection and Classification model leverages the following features:

Real-Time Detection: Built on the YOLOv8 architecture, this model is optimized for real-time detection tasks, making it highly suitable for in-field agricultural monitoring where rapid response times are critical.

Multi-Class Classification: With 46 plant classes, the model provides an extensive coverage of crop types, which allows for a diverse range of applications beyond simple leaf detection. Each detected leaf is assigned to one of the classes, which could facilitate tasks like mixed cropping or monitoring crop health across different plant species.

Potential for Water Stress Detection: Although originally trained for general leaf classification, the model's adaptability enables it to be applied to more specialized tasks, such as detecting water stress. In this context, the model can be used to analyze visual signs of stress on potato leaves, such as discoloration, wilting, and texture changes, that are indicative of water deficiency.

Chapter 4

Results and Discussions

4.1 Evaluation Metrics

Evaluation metrics play a crucial role in measuring the effectiveness of machine learning models, particularly in fields like object detection and image segmentation. The selection of the appropriate metric allows for a better understanding of how well the model performs, identifying areas where it excels and areas for improvement. This section focuses on key evaluation metrics such as **Precision**, **Recall**, **mAP**, and **IoU**.

Precision Precision refers to the ratio of correct positive predictions to the total positive predictions generated by the model. It reflects how accurate the model's positive predictions are. In object detection, precision is crucial for reducing false positives, which occur when the model incorrectly identifies an object.

The formula for precision is given in Equation 4.1:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

Where:

- **TP** (True Positives): Correctly predicted positive instances (e.g., correctly identifying water-stressed crops).
- **FP** (False Positives): Instances where the model incorrectly predicted a positive outcome (e.g., misclassifying healthy crops as stressed).

In agriculture, particularly in detecting water-stressed crops, high precision ensures that the majority of plants identified as stressed actually require intervention, thereby avoiding false detections of healthy crops as stressed. This is critical, as incorrect identification could lead to unnecessary actions like over-irrigation, which

not only wastes water but can also damage crops or increase operational costs. False positives in this context—where healthy plants are identified as stressed—can result in overwatering, leading to resource inefficiency and, potentially, crop damage. In water-scarce regions, the need for water conservation is paramount. A model with high precision ensures that water is used efficiently by preventing false alarms and unnecessary irrigation. This improves the overall sustainability of farming practices by optimizing water usage while maintaining crop health. Achieving high precision in water stress detection presents several challenges due to the difficulty in visually differentiating between healthy and mildly stressed plants. Environmental variables, including fluctuating lighting, shadows, and overlapping plant canopies, add further complexity to this differentiation. As a result, sophisticated models and high-quality data are required to accurately detect and classify plant stress. The precision of a model is strongly influenced by the quality of the training data and the accuracy of the annotations. High-quality, well-annotated datasets are essential for the model to effectively learn the differences between healthy and stressed crops. Properly labeled data ensures the model is trained on accurate examples, which directly impacts its performance in real-world scenarios. This is particularly important for detecting varying levels of stress and applying the correct interventions.

Recall Recall, sometimes referred to as Sensitivity or the True Positive Rate, indicates the ratio of accurately predicted positive cases to the total number of actual positive cases. In the context of water stress detection, recall indicates the model's ability to identify all water-stressed crops. The formula for recall is shown in Equation 4.2:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

Where:

- **TP** (True Positives): The number of correctly detected stressed crops.
- **FN** (False Negatives): The number of stressed crops that were not detected by the model.

In water stress detection, high recall is crucial as it ensures that most, if not all, stressed crops are identified. Missing water-stressed plants can lead to under-irrigation, causing harm to crop health and reducing yield. Furthermore, high recall ensures that all areas needing irrigation are identified, allowing for precise water management. Especially in large-scale farming, missed detections (false negatives) can significantly impact crop productivity. Thus, high recall ensures effective irrigation and timely interventions, which are critical for maintaining crop health.

Mean Average Precision (mAP) is a commonly used metric for assessing the performance of object detection models. In the context of water stress detection, it offers an overall assessment of the model's capability to maintain a balance between precision and recall across various Intersection over Union (IoU) thresholds. Calculating mAP involves computing the Average Precision (AP) for each class by calculating the area under the precision-recall curve, which plots precision against recall at varying confidence thresholds. Once the AP for each class is computed, the mAP is derived by averaging the AP values across all classes. This results in a single scalar value summarizing the overall performance of the model.

Different variations of mAP are used, depending on the IoU thresholds:

- **mAP@0.5 (mAP50)**: This metric calculates the average precision at an IoU threshold of 0.5, indicating the extent to which the predicted bounding boxes align with the ground truth boxes by a minimum of 50%..
- **mAP@0.5:0.95 (mAP50-95)**: This metric evaluates average precision across multiple IoU thresholds, from 0.5 to 0.95 in 0.05 increments. It provides a more thorough evaluation of the model's performance across different levels of IoU.

In water stress detection, mAP helps evaluate the model's capability to accurately localize and classify stressed plants across varying IoU thresholds, ensuring that regions requiring attention are correctly identified and irrigated.

Confusion Matrix The **Confusion Matrix** is a valuable tool for breaking down the model's performance by presenting the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). It provides a detailed visualization of the model's performance and is particularly useful for balancing precision and recall in water stress detection. The confusion matrix is structured as follows in Table 4.1:

	Predicted Stressed	**Predicted Healthy**
Actual Stressed	TP	FN
Actual Healthy	FP	TN

Table 4.1: Confusion Matrix for Water Stress Detection

Where:

- **TP** (True Positives): The number of correctly predicted water-stressed crops.
- **TN** (True Negatives): The number of healthy crops correctly identified as not stressed.
- **FP** (False Positives): The number of healthy crops incorrectly classified as stressed.

- **FN** (False Negatives): The number of stressed crops that the model failed to detect.

In water stress detection, analyzing the confusion matrix is essential to strike a balance between minimizing false negatives (missed stressed crops) and false positives (incorrectly flagged healthy crops). Understanding the model's performance across these metrics is key to optimizing irrigation systems and ensuring the health and productivity of crops.

4.2 Experimental Results

4.2.1 Results for Initial Approach

Although segmentation and vegetation indices were initially explored as potential approaches for detecting water-stressed regions in plants, the experimentation with these methods did not reach full completion. The segmentation approach combined color-based and threshold-based techniques to separate stressed plants from healthy ones, followed by calculating the NDVI index to assess plant health.

The segmentation results provide qualitative insights into the potential of these methods. However, quantitative metrics such as precision, recall, and Intersection over Union (IoU) were not calculated for this approach. Preliminary visual results are shown in Appendix A, where the segmented regions of the plants are highlighted along with NDVI calculations.

Despite not having quantitative evaluation, these preliminary results suggest that integrating NDVI with segmentation could be a viable approach for assessing water stress in crops. Further refinement and comprehensive evaluation would be required to fully validate the performance of this method.

4.2.2 YOLO Results

To maintain a clear and organized presentation of the results, only the most significant findings are highlighted in visual form, while the remaining configurations are summarized in tables. This approach prevents the results section from becoming overwhelmed with repetitive data, allowing a focus on key outcomes.

Hyperparameter Tuning for YOLOv8n

Various combinations of learning rates, batch sizes, and image sizes were tested on the YOLOv8n model across multiple image types, including RGB, RGN, RGE, and RGREN and RGBN. Specifically, the following configurations were explored:

Learning Rate	Batch Size	Image Size	Image Type
0.1	16	640	All
0.01	16	640	All
0.001	16	640	All
0.01	32	640	All
0.01	16	416	All except RGB images.

Table 4.2: YOLOv8n hyperparameter configurations tested on different image types (RGB, RGN, RGE, RGREN, and RGBN).

These configurations were designed to assess the impact of each parameter on model performance. The best configurations, determined by the highest mAP, precision, and recall, are presented with full visual data, including training loss curves, confusion matrices, and detailed metric tables. The remaining configurations are compiled into a summary table for comparative analysis. IoU is 0.7 for all the experiments.

Hyperparameter Tuning for YOLOv8s

The best-performing configuration from YOLOv8n was then applied to the YOLOv8s model to evaluate its accuracy and potential for overfitting. Results Presentation For the configurations demonstrating, full visual results (training plots, confusion matrices, and performance metrics) are provided. Other configurations are summarized in a table, displaying precision, recall, and mAP values across the different learning rates, batch sizes, and image sizes.

RGB Images

The table 4.3 shows the performance of the YOLOv8n model on the Augmented RGB images in potato dataset. It includes metrics for overall performance, as well as specific results for detecting healthy and stressed. Additionally, the table lists the model's training parameters, including the number of epochs, batch size, and image size used during training. It is observed that epochs more than 100 will cause the model to overfit on the test datasest.

Performance Metrics for YOLOv8n Configurations						
Learning Rate	**Batch Size**	**Image Size**	**Precision**	**Recall**	**mAP50**	**mAP50-95**
0.1	16	640	0.8233	0.7037	0.8071	0.6614
0.01	16	640	0.8150	0.7806	0.8434	0.6945
	16	416	0.8325	0.7302	0.8530	0.6397
	32	640	0.8125	0.7791	0.8701	0.6351
0.001	16	640	0.8175	0.7693	0.8684	0.6441

Table 4.3: YOLOv8n performance on RGB Images with Different hyper parametrs settings.

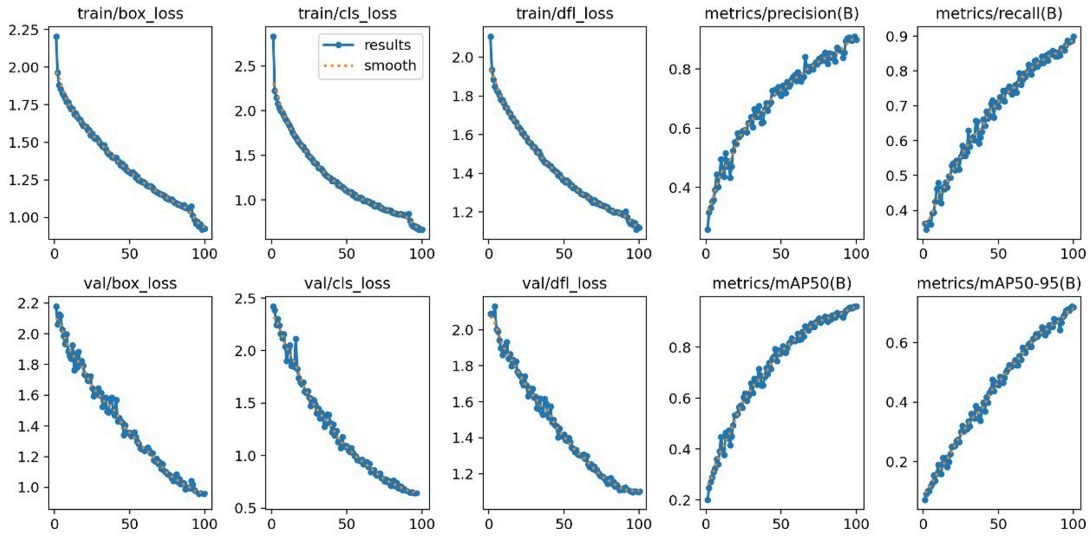


Figure 4.1: Training and Validation Loss and Metrics Plots

The training plots 4.1 represent the performance metrics and losses obtained during the training and validation phases of a YOLOv8 object detection model. These metrics help in tracking how well the model learns over the course of training epochs and how effectively it generalizes to unseen validation data. The plots show trends across 100 epochs, highlighting both the loss reduction and the improvement in detection metrics as the model refines its understanding of the input data.

- Train Box Loss: This plot shows how the bounding box regression loss decreases over time during the training phase. A downward trend in this loss suggests that the model is becoming better at precisely predicting object locations within the images. As the epochs increase, the box loss continues to decrease, signifying better localization of objects.

- Train Classification Loss: This plot visualizes the classification loss during training. This loss measures how well the model classifies objects into the correct categories (e.g., healthy, stressed). A continuous decrease indicates the model is becoming better at distinguishing between different object classes as training progresses.
- Train DFL Loss: The DFL loss (Distribution Focal Loss) is specific to YOLOv8 and is used to improve the model's confidence in its bounding box predictions. This loss also shows a decreasing trend, which means the model is learning to make more confident predictions about the exact positions of object boundaries.
- Train Precision (B): This plot tracks the precision during training. Precision represents the percentage of identified positives (detections) that were indeed accurate. As precision increases over epochs, the model becomes more selective and accurate, minimizing false positives.
- Train Recall (B): This plot displays the recall during training. Recall measures how many of the actual positive cases (i.e., true objects in the image) the model correctly identified. As recall increases, the model gets better at detecting more objects without missing them.
- Validation Box Loss: Similar to train box loss, this plot shows the bounding box regression loss during the validation phase. The fact that it decreases steadily indicates that the model generalizes well to unseen data in terms of object localization.
- Validation Classification Loss: This plot tracks the classification loss during the validation phase. A smooth decrease indicates that the model is improving its ability to classify objects on the validation set, suggesting better generalization to new data.
- Validation DFL Loss: Similar to the training DFL loss, this plot represents the confidence in bounding box predictions during validation. A decreasing DFL loss indicates better confidence in localization predictions, even on validation data.
- Validation mAP50: This plot shows the mean Average Precision (mAP) at 50% Intersection over Union (IoU) for the validation set. mAP50 is a crucial metric for object detection, measuring how well the model's predicted bounding boxes match the ground truth. The steady increase indicates improved performance in detecting and localizing objects correctly.
- Validation mAP50-95: This plot tracks the mean Average Precision across a range of IoU thresholds (from 50% to 95%). A higher mAP50-95 means the

model is capable of making precise bounding box predictions, even at stricter IoU thresholds. The increasing trend suggests that the model is improving its general detection performance.

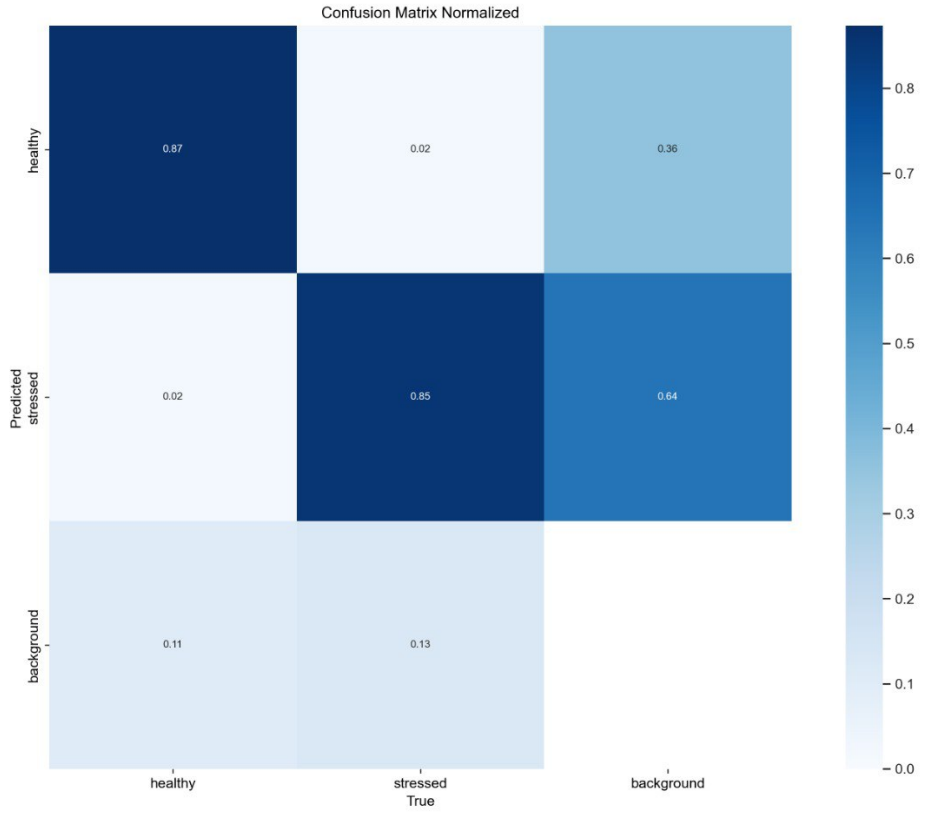


Figure 4.2: Confusion Matrix for RGB images running 100 epochs using yolov8n

The above confusion matrix illustrates the performance of the model in predicting healthy, stressed, and background classes. It shows the true positive, false positive, and false negative rates for each class, providing insight into the model's accuracy and misclassification rates. For more insight, Table 4.4 explains the meaning of this confusion matrix.

Table 4.4: Confusion Matrix Interpretation for Healthy, Stressed, and Background Classes

Class	**True Positives (TP)**	**False Positives (FP)**	**False Negatives (FN)**
healthy	87%	38%	13%
stressed	85%	66%	36%

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	2147	0.941	0.875	0.931	0.727
Healthy	121	858	0.943	0.892	0.940	0.737
Stressed	121	1289	0.939	0.858	0.922	0.711
Epochs						100
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.5: YOLOv8s performance (100 epochs) on RGB Images.

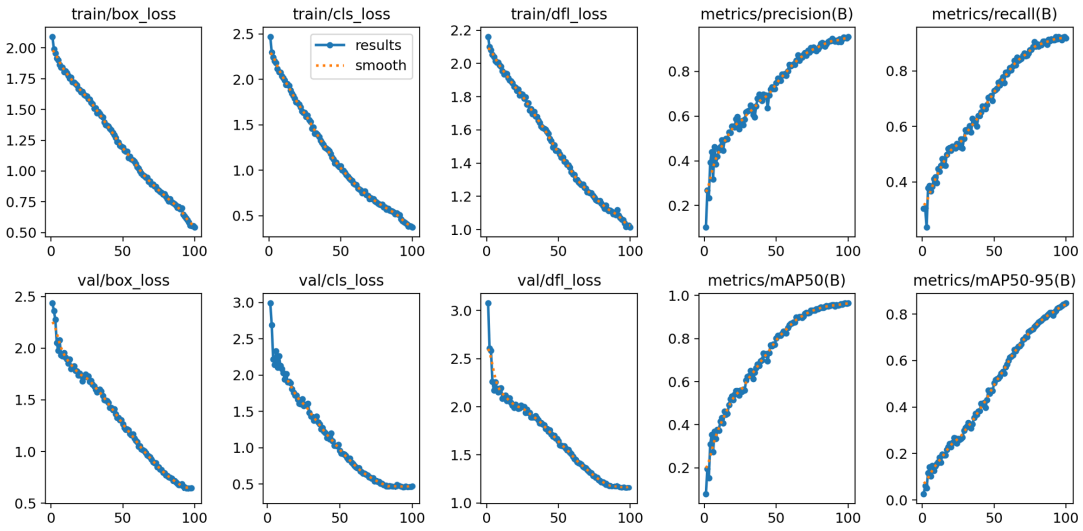


Figure 4.3: Training and Validation Loss and Metrics Plots

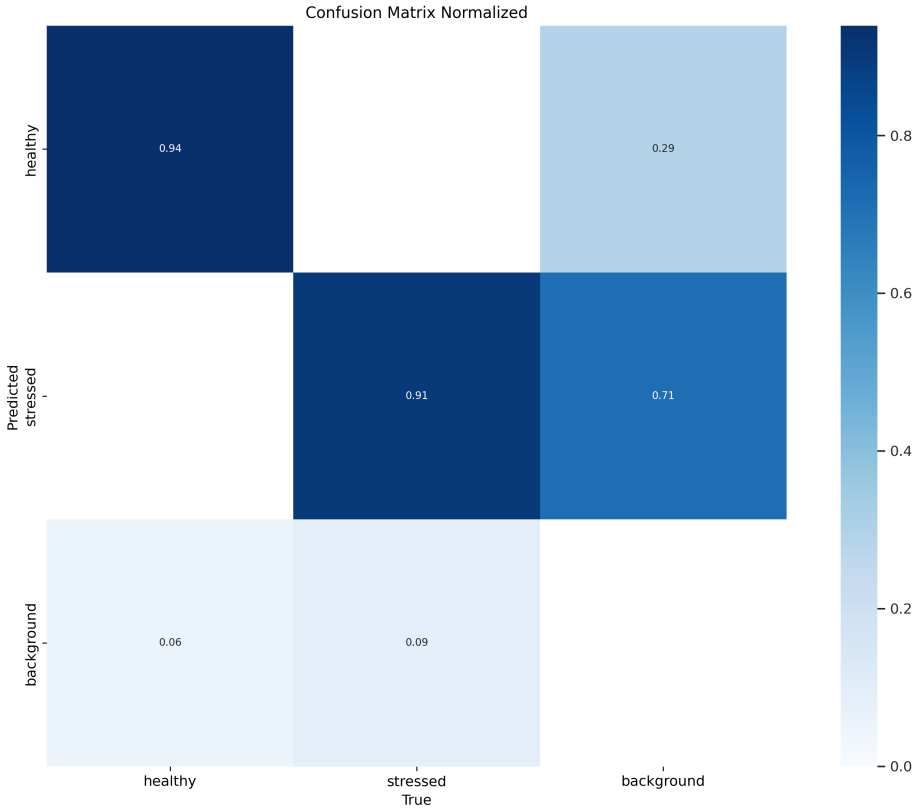


Figure 4.4: Confusion Matrix for RGB images running 100 epochs using yolov8s

RGN Images

Performance Metrics for YOLOv8n Configurations						
Learning Rate	**Batch Size**	**Image Size**	**Precision**	**Recall**	**mAP50**	**mAP50-95**
0.1	16	640	0.68491	0.57082	0.60114	0.46029
0.01	16	640	0.68385	0.60047	0.68481	0.47548
	16	416	0.68207	0.56848	0.63511	0.30841
	32	640	0.78610	0.69012	0.76303	0.52621
0.001	16	640	0.78274	0.66736	0.73401	0.57437

Table 4.6: YOLOv8n performance (100 epochs) on RGN Images with Different hyper parametrs settings.

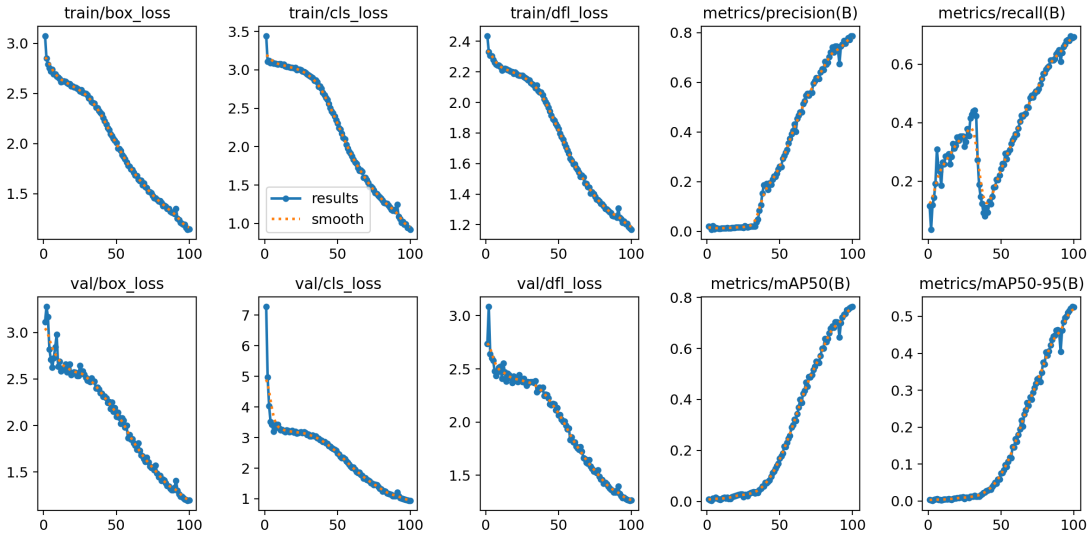


Figure 4.5: Training and Validation Loss and Metrics Plots

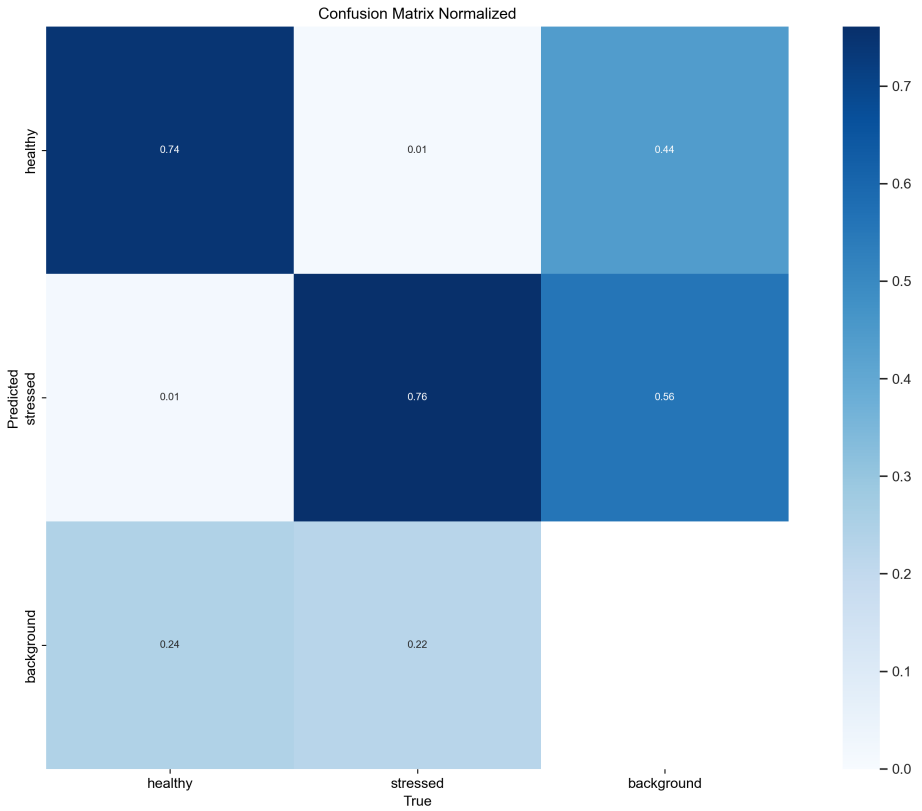


Figure 4.6: Confusion Matrix for RGN images running 100 epochs using yolov8n

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	1699	0.833	0.789	0.865	0.565
Healthy	121	664	0.831	0.777	0.862	0.562
Stressed	121	1035	0.834	0.802	0.869	0.569
Epochs						80
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.7: YOLOv8s performance (80 epochs) on RGN Images.

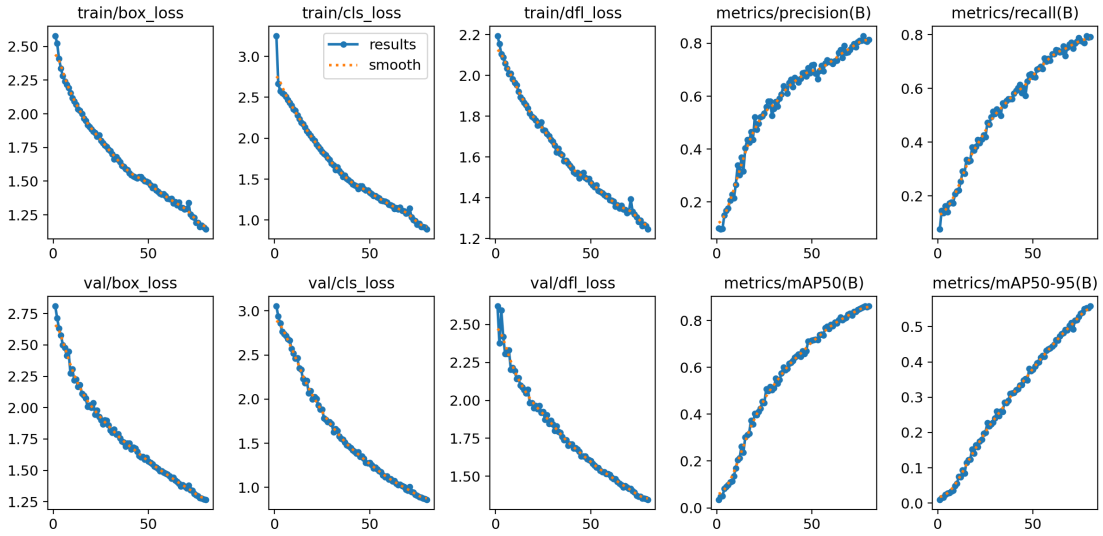


Figure 4.7: Training and Validation Loss and Metrics Plots

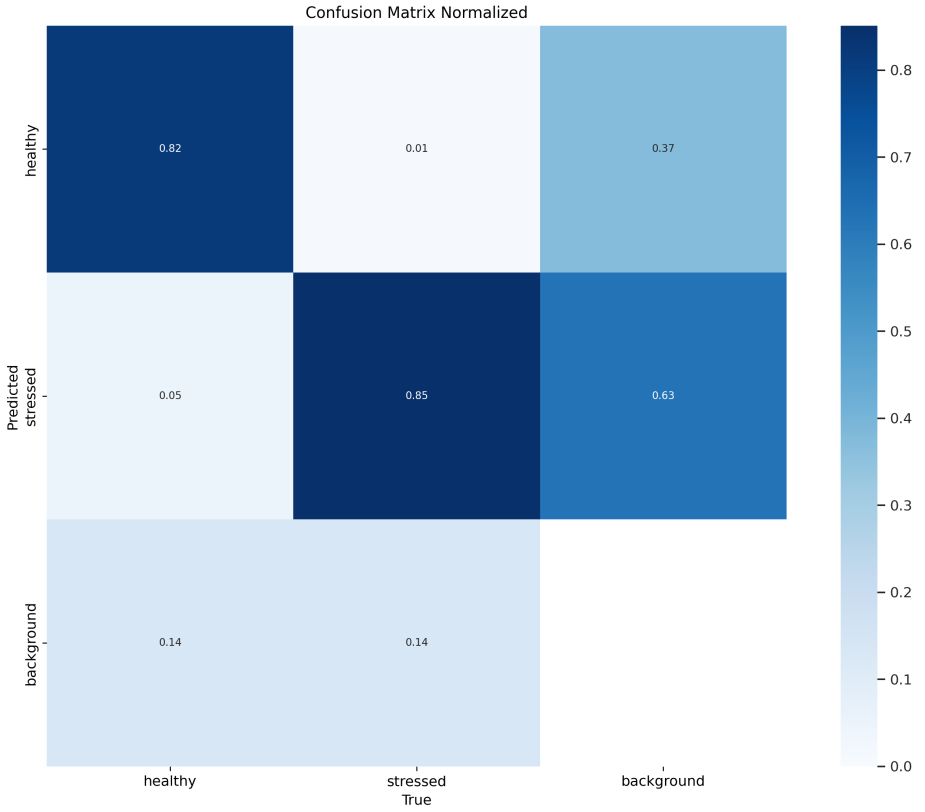


Figure 4.8: Confusion Matrix for RGN images running 80 epochs using yolov8s

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	1699	0.906	0.875	0.931	0.670
Healthy	121	664	0.908	0.87	0.934	0.668
Stressed	121	1035	0.902	0.9	0.924	0.663
Epochs						100
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.8: YOLOv8s performance (100 epochs) on RGN Images.

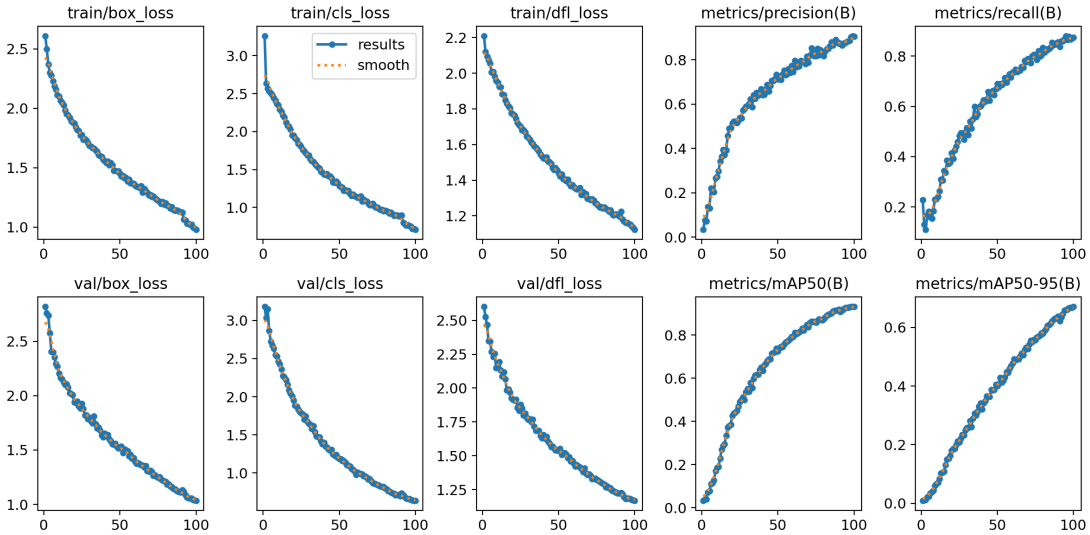


Figure 4.9: Training and Validation Loss and Metrics Plots

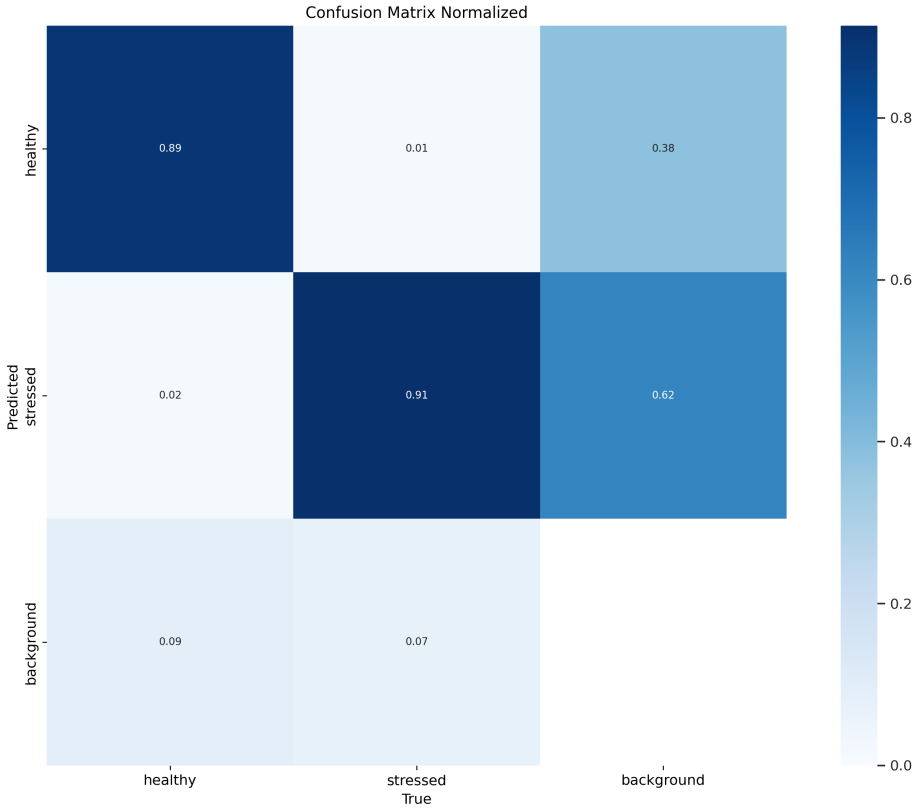


Figure 4.10: Confusion Matrix for RGN images running 100 epochs using yolov8s

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	1699	0.971	0.959	0.986	0.841
Healthy	121	664	0.972	0.959	0.987	0.847
Stressed	121	1035	0.969	0.958	0.986	0.834
Epochs						100
Batch Size						32
Image Size						640
Model Size						Pretrained Hugging Face
Learning Rate						0.01

Table 4.9: Pretrained Hugging Face performance (100 epochs) on RGN Images.

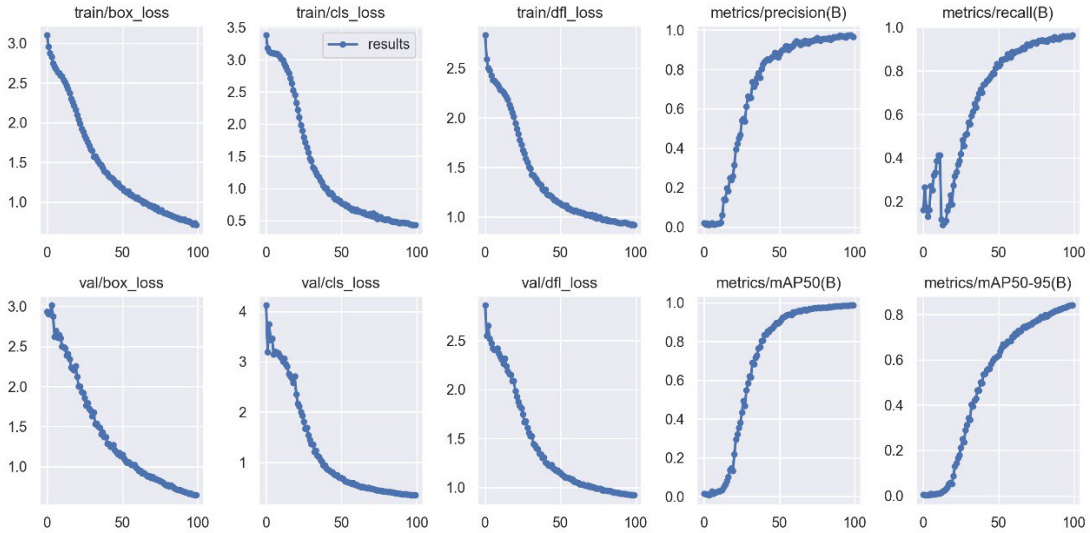


Figure 4.11: Training and Validation Loss and Metrics Plots

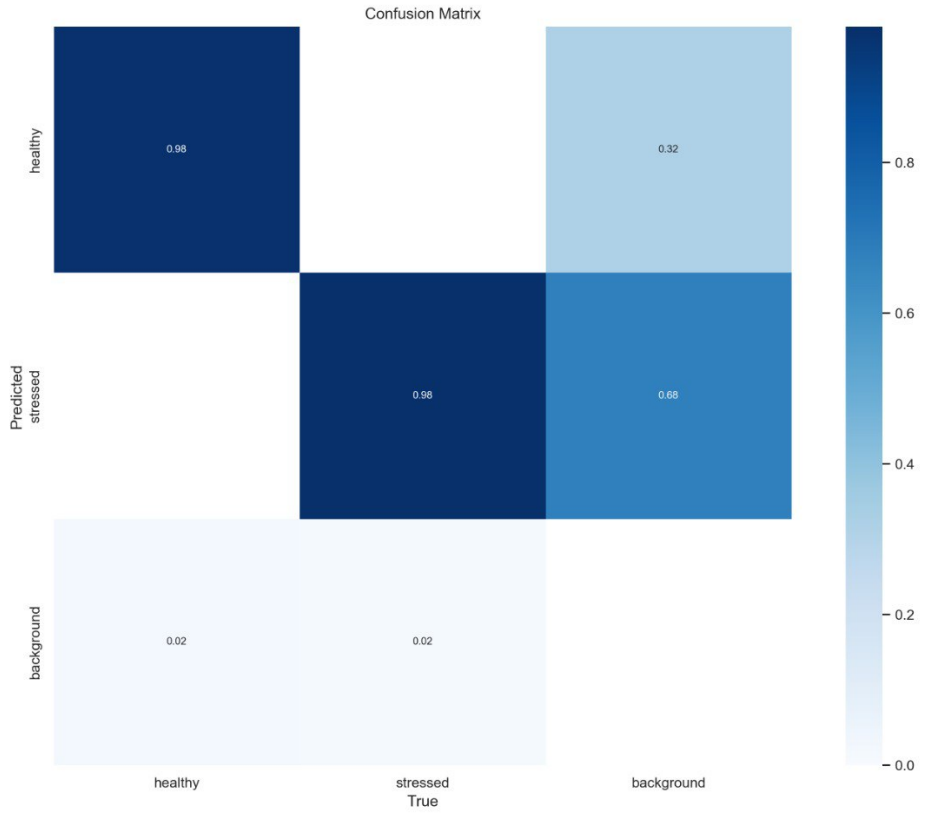


Figure 4.12: Confusion Matrix for RGN images running 100 epochs using Pre-trained mode Hugging Face

RGE Images

Performance Metrics for YOLOv8n Configurations						
Learning Rate	**Batch Size**	**Image Size**	**Precision**	**Recall**	**mAP50**	**mAP50-95**
0.1	16	640	0.6134	0.5712	0.6034	0.2862
0.01	16	640	0.60742	0.4652	0.55304	0.27973
	16	416	0.62048	0.44619	0.49667	0.24579
	32	640	0.63197	0.52995	0.55874	0.31337
0.001	16	640	0.60429	0.46475	0.50129	0.28564

Table 4.10: YOLOv8n performance (100 epochs) on RGE Images with Different hyper parametrs settings.

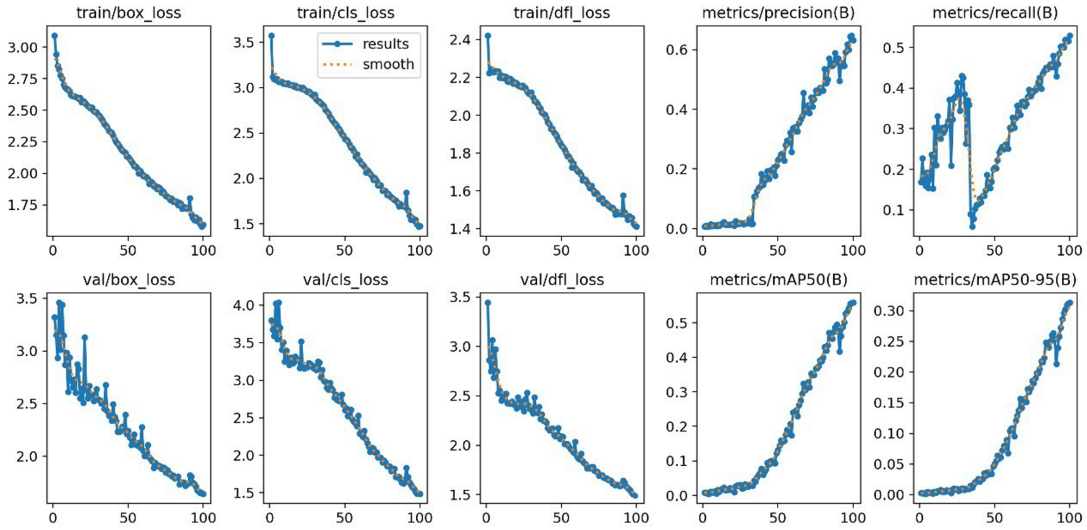


Figure 4.13: Training and Validation Loss and Metrics Plots

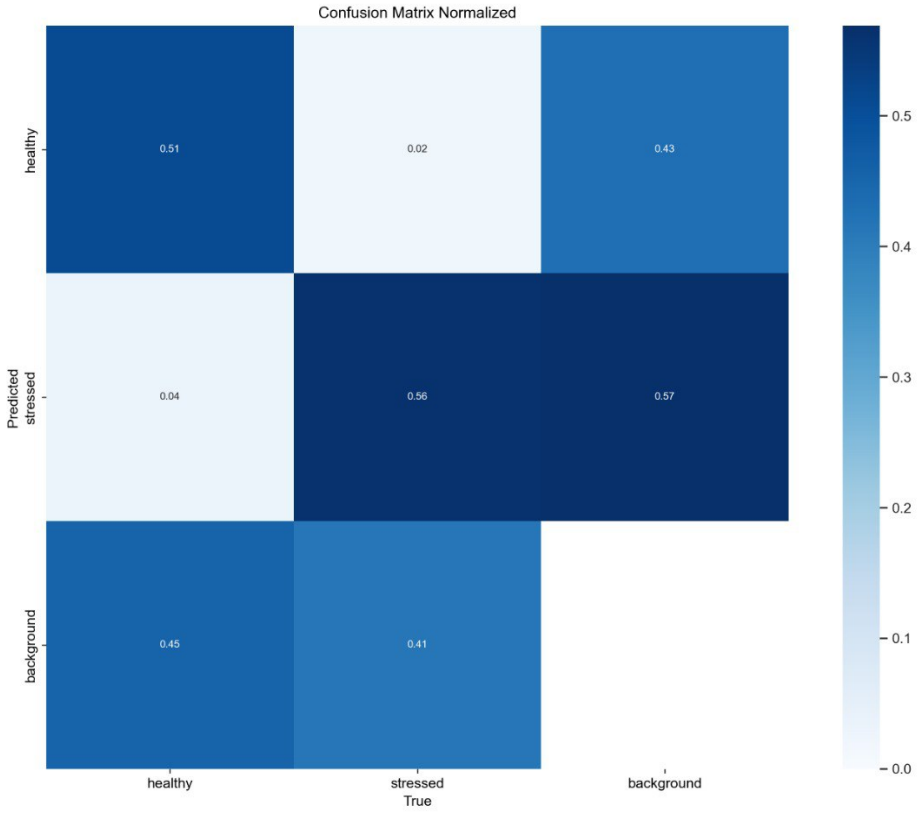


Figure 4.14: Confusion Matrix for RGE images running 100 epochs using yolov8n

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	1699	0.865	0.806	0.829	0.649
Healthy	121	664	0.858	0.806	0.824	0.645
Stressed	121	1035	0.870	0.818	0.826	0.652
Epochs						80
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.11: YOLOv8s performance (80 epochs) on RGE Images.

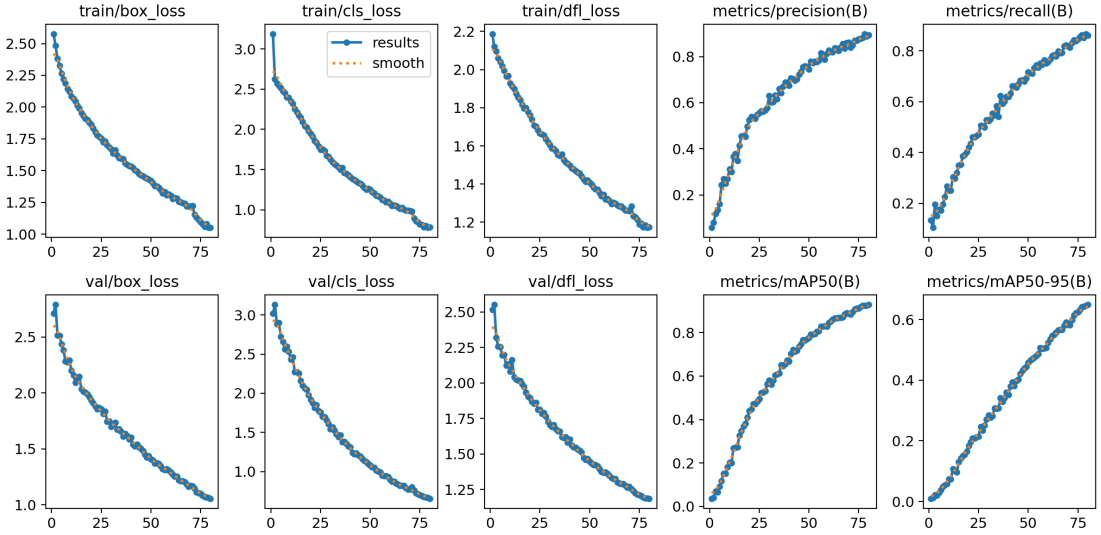


Figure 4.15: Training and Validation Loss and Metrics Plots

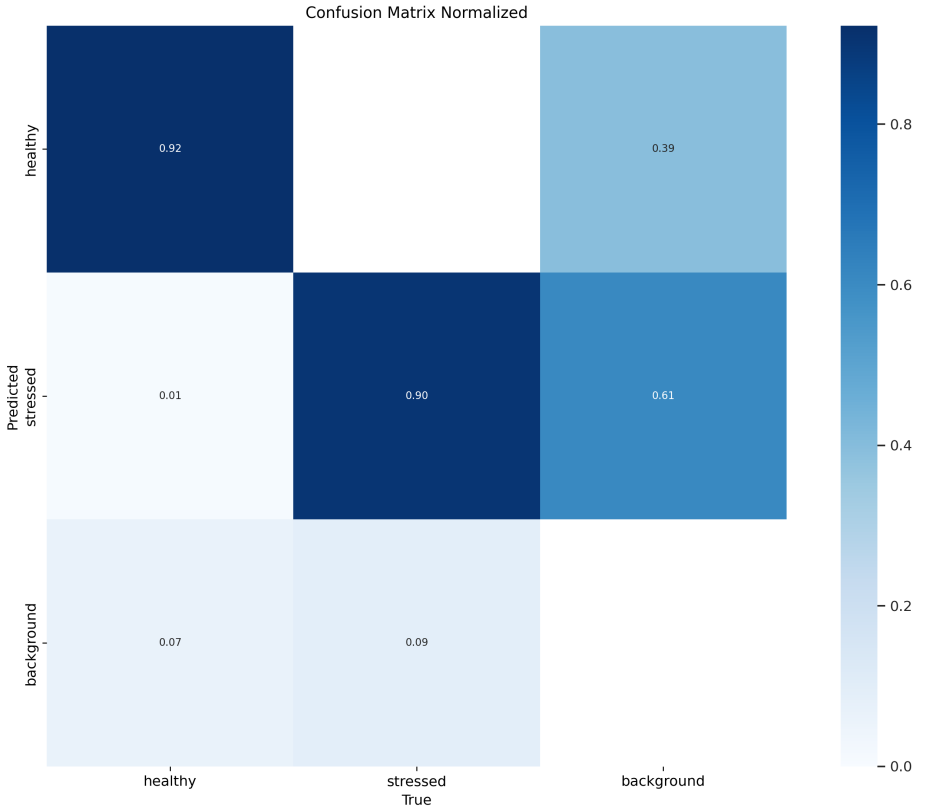


Figure 4.16: Confusion Matrix for RGE images running 80 epochs using yolov8s

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	1699	0.914	0.902	0.952	0.707
Healthy	121	664	0.913	0.900	0.952	0.705
Stressed	121	1035	0.911	0.901	0.950	0.698
Epochs						100
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.12: YOLOv8s performance (100 epochs) on RGE Images.

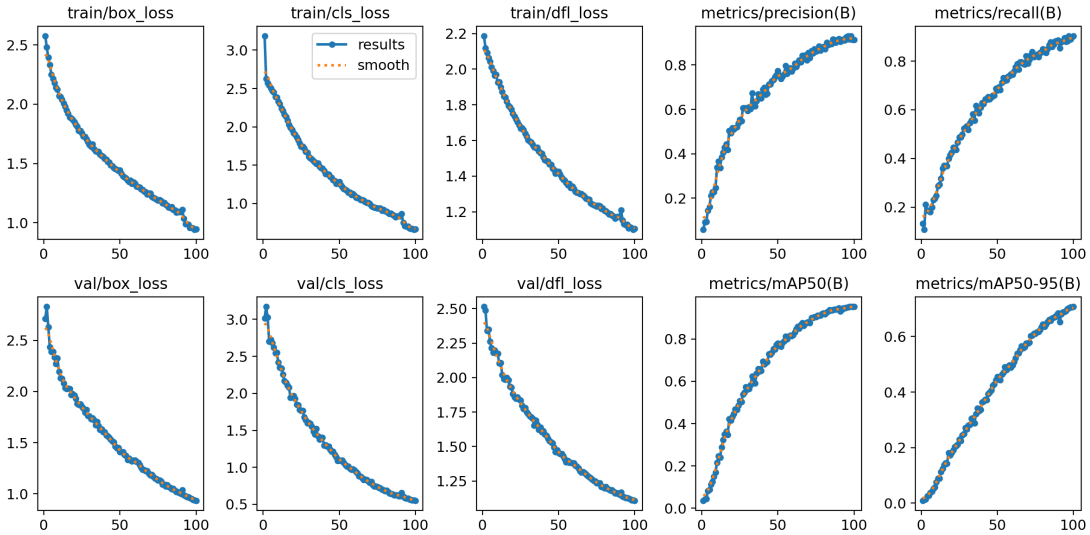


Figure 4.17: Training and Validation Loss and Metrics Plots

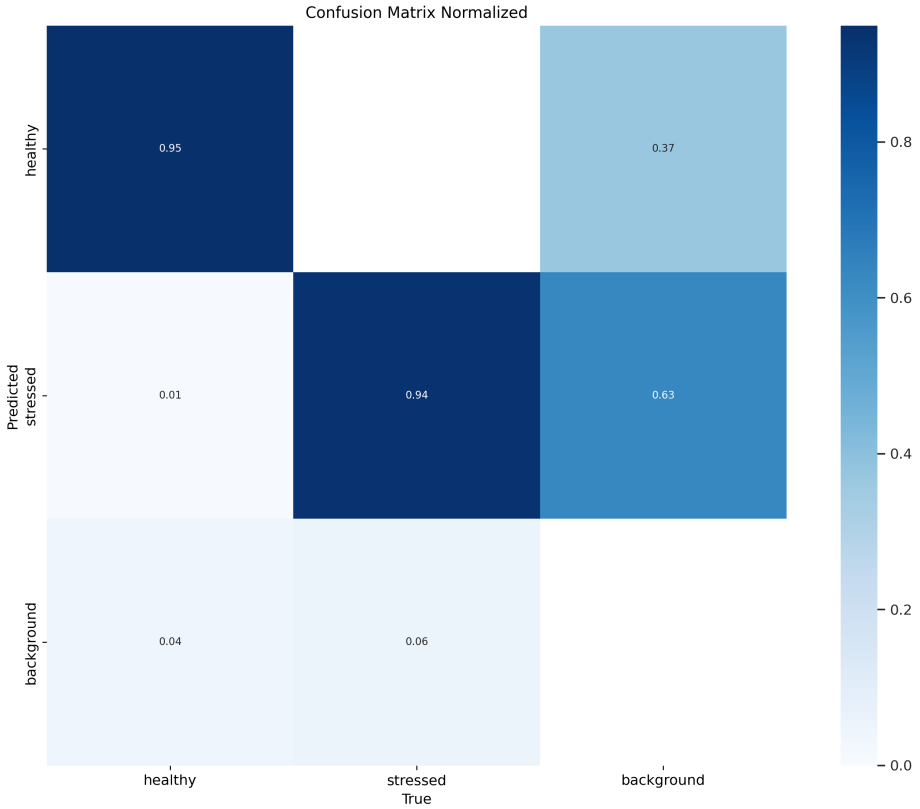


Figure 4.18: Confusion Matrix for RGE images running 100 epochs using yolov8s

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	1699	0.959	0.963	0.990	0.884
Healthy	121	664	0.959	0.964	0.990	0.884
Stressed	121	1035	0.959	0.961	0.990	0.884
Epochs						100
Batch Size						32
Image Size						640
Model						Pretrained Hugging Face
Learning Rate						0.01

Table 4.13: Pretrained Hugging Face performance (100 epochs) on RGE Images.

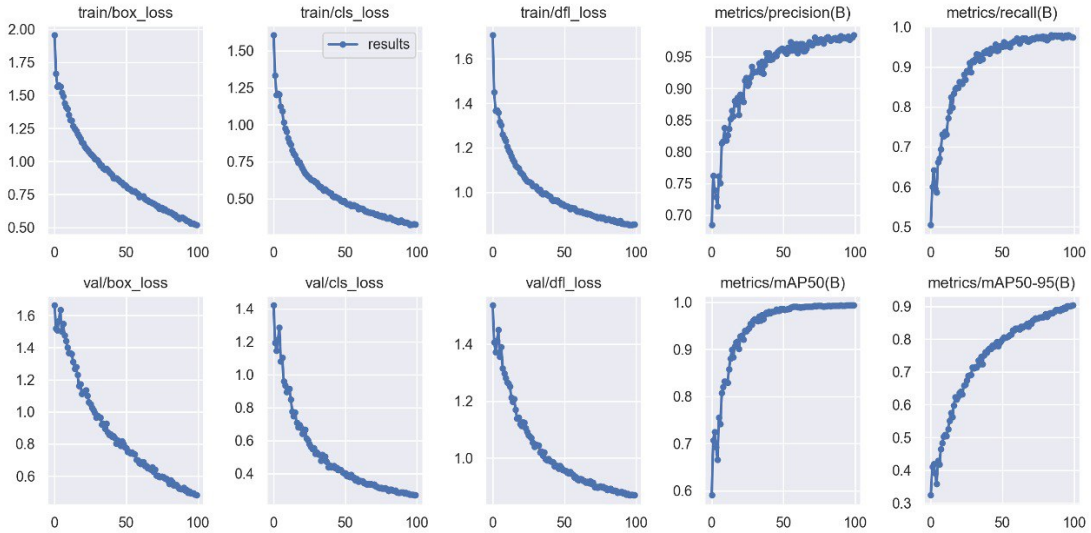


Figure 4.19: Training and Validation Loss and Metrics Plots

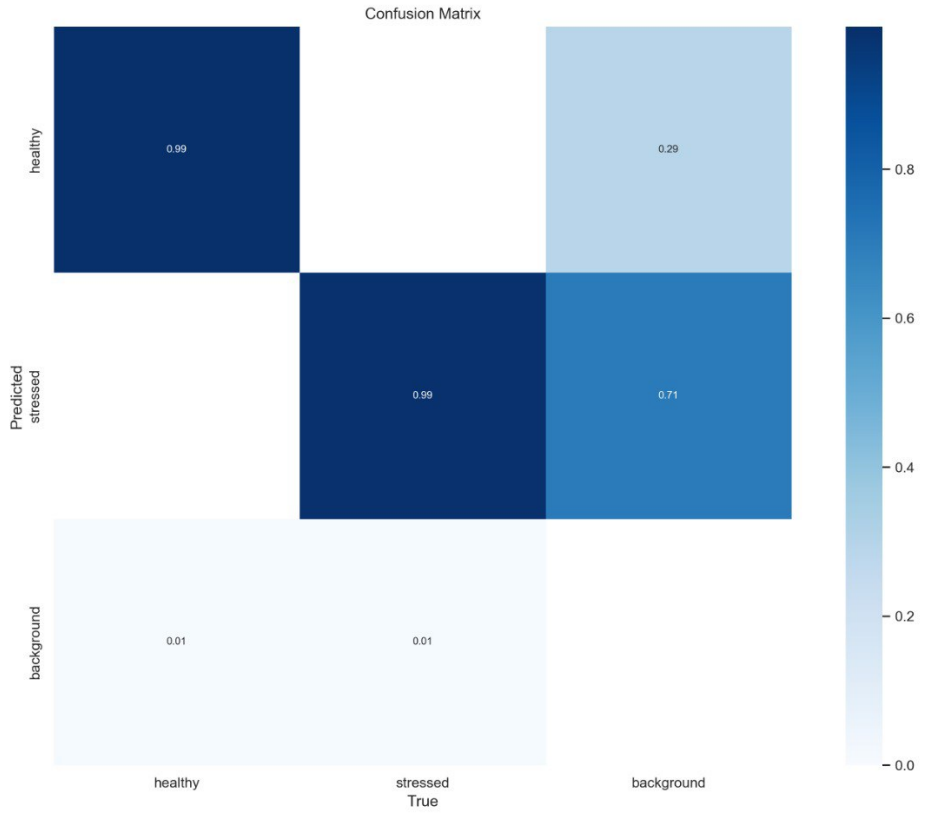


Figure 4.20: Confusion Matrix for RGE images running 100 epochs using Pre-trained Hugging Face

RGREN Images

Performance Metrics for YOLOv8n Configurations						
Learning Rate	**Batch Size**	**Image Size**	**Precision**	**Recall**	**mAP50**	**mAP50-95**
0.1	16	640	0.65688	0.55173	0.6034	0.34751
0.01	16	640	0.65191	0.53955	0.58495	0.33581
	16	416	0.67254	0.49702	0.55304	0.27973
	32	640	0.70934	0.60082	0.65754	0.39387
0.001	16	640	0.65688	0.55173	0.6034	0.34751

Table 4.14: YOLOv8n performance (100 epochs) on RGREN Images with Different hyper parametrs settings.

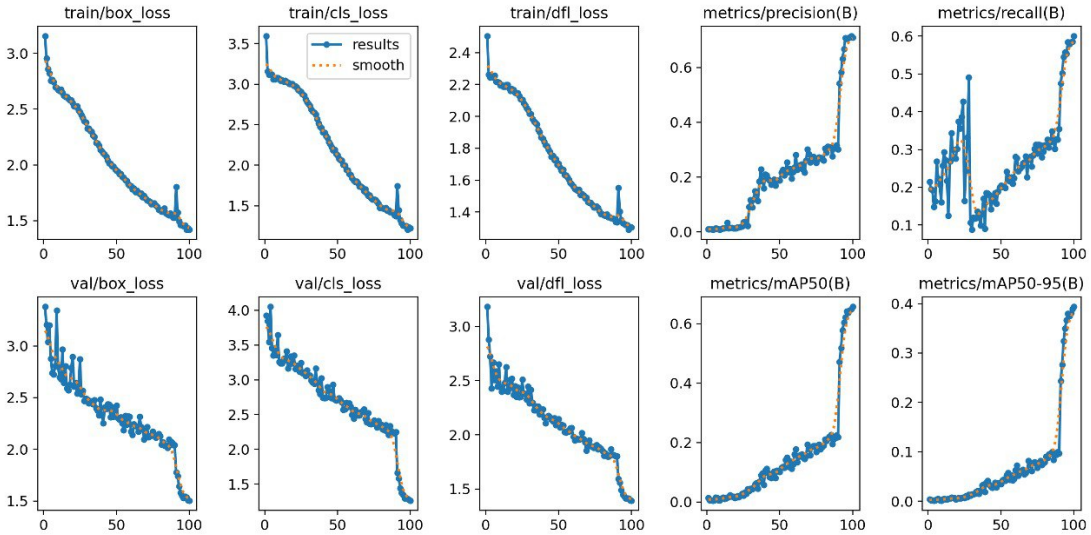


Figure 4.21: Training and Validation Loss and Metrics Plots

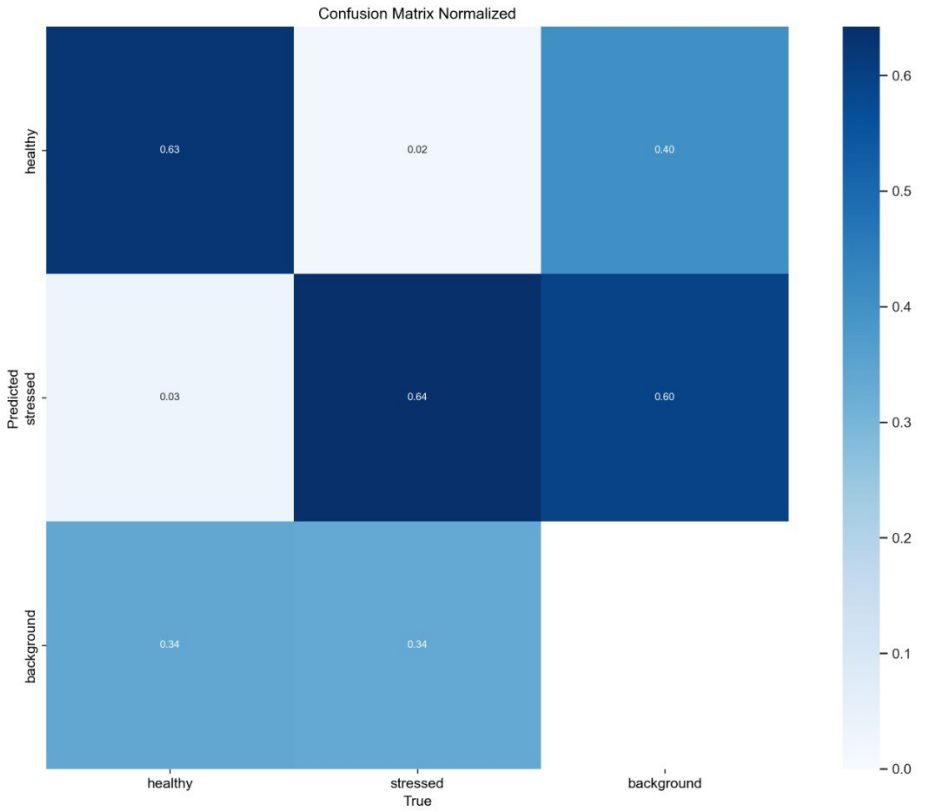


Figure 4.22: Confusion Matrix for RGREN images running 100 epochs using yolov8n

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	4006	0.823	0.744	0.782	0.505
Healthy	121	1679	0.813	0.695	0.781	0.512
Stressed	121	2327	0.834	0.696	0.783	0.498
Epochs						150
Batch Size						16
Image Size						640
Model Size						yolov8n
Learning Rate						0.01

Table 4.15: YOLOv8n performance (150 epochs) on RGREN Images.

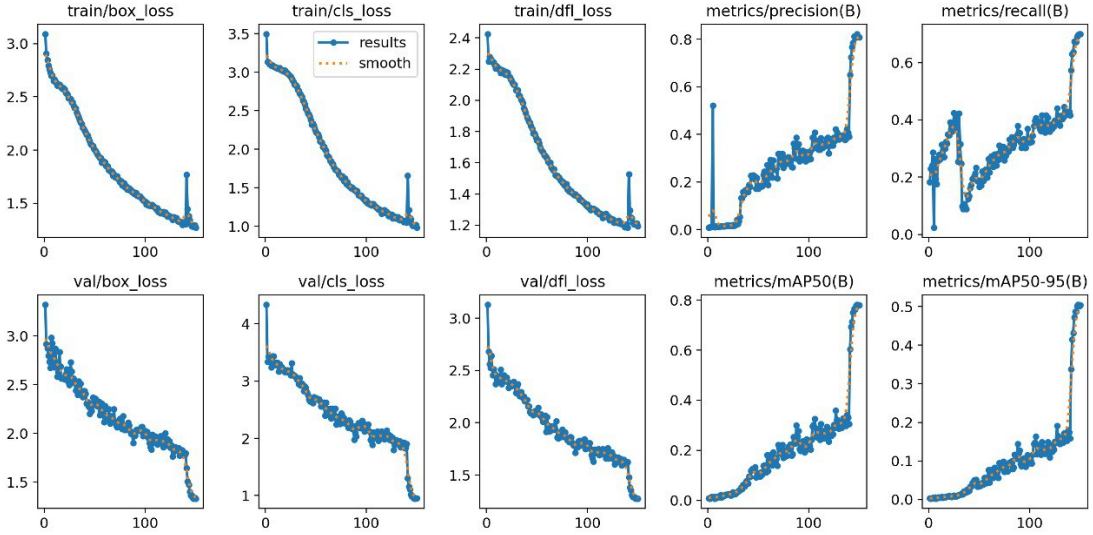


Figure 4.23: Training and Validation Loss and Metrics Plots

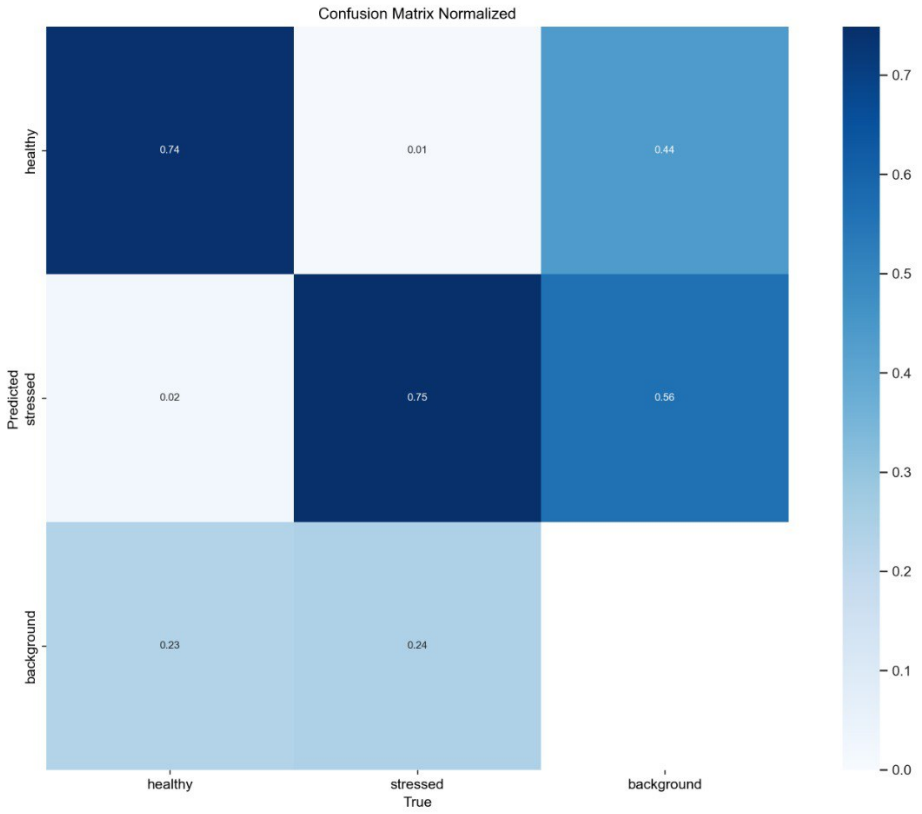


Figure 4.24: Confusion Matrix for RGREN images with 150 epochs

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	4006	0.762	0.688	0.758	0.515
Healthy	121	1679	0.778	0.693	0.771	0.529
Stressed	121	2327	0.745	0.683	0.744	0.502
Epochs						80
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.16: YOLOv8s performance (80 epochs) on RGREN Images.

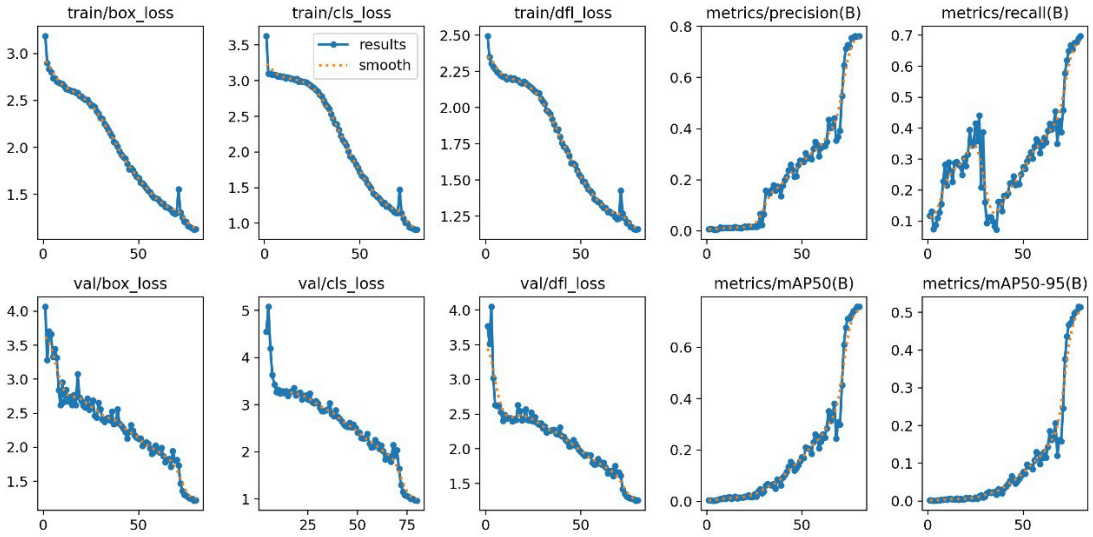


Figure 4.25: Training and Validation Loss and Metrics Plots

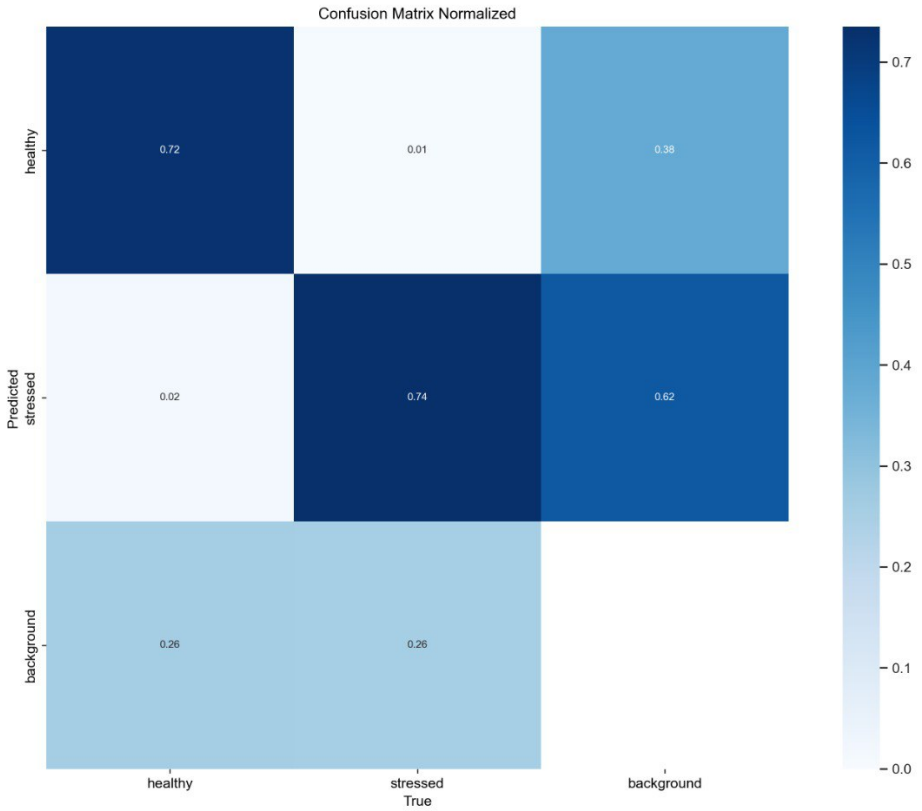


Figure 4.26: Confusion Matrix for RGREN images running 80 epochs using yolov8s

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	4006	0.856	0.694	0.807	0.507
Healthy	121	1679	0.854	0.693	0.818	0.514
Stressed	121	2327	0.847	0.694	0.808	0.501
Epochs						100
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.17: YOLOv8s performance (100 epochs) on RGREN Images.

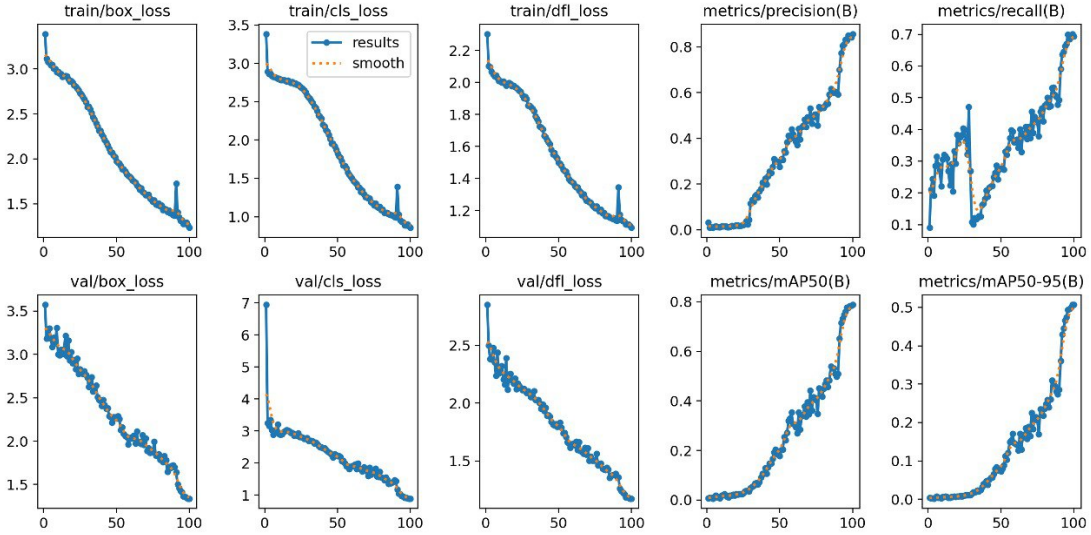


Figure 4.27: Training and Validation Loss and Metrics Plots

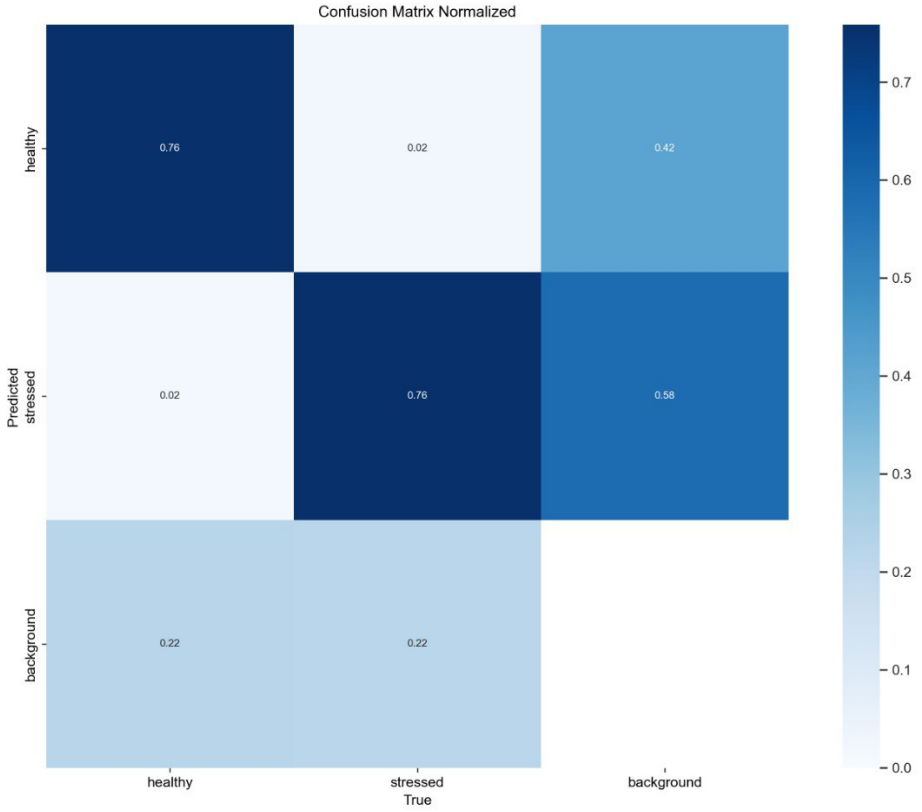


Figure 4.28: Confusion Matrix for RGREN images running 100 epochs using yolov8s

RGBN Images

Performance Metrics for YOLOv8n Configurations						
Learning Rate	**Batch Size**	**Image Size**	**Precision**	**Recall**	**mAP50**	**mAP50-95**
0.1	16	640	0.64797	0.55186	0.61021	0.36842
0.01	16	640	0.6922	0.6	0.6563	0.3724
	16	416	0.6541	0.4925	0.5441	0.2612
	32	640	0.71254	0.60112	0.64854	0.41324
0.001	16	640	0.65718	0.55281	0.60431	0.34812

Table 4.18: YOLOv8n performance (100 epochs) on RGBN Images with Different hyper parametrs settings.

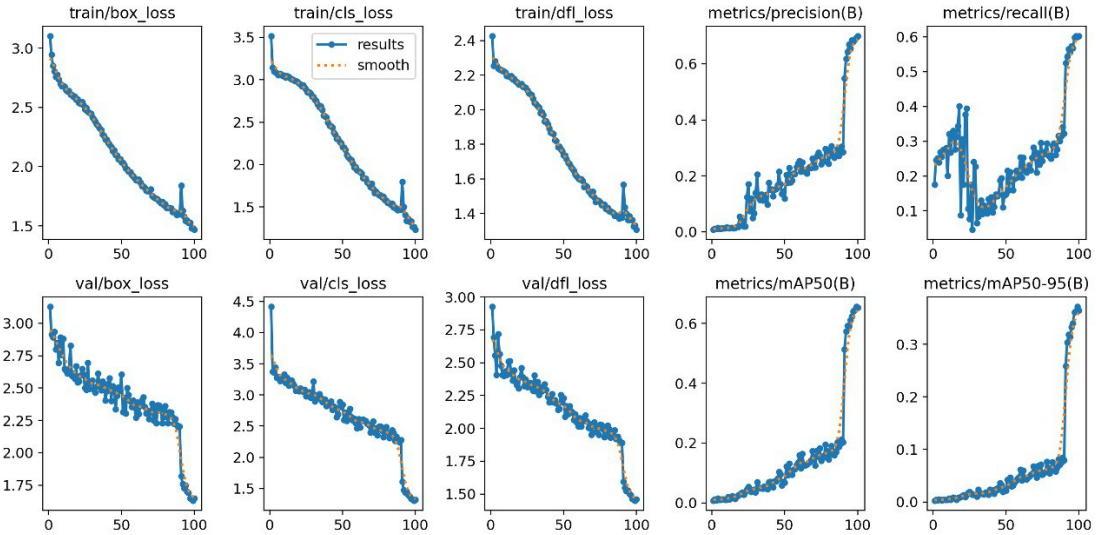


Figure 4.29: Training and Validation Loss and Metrics Plots

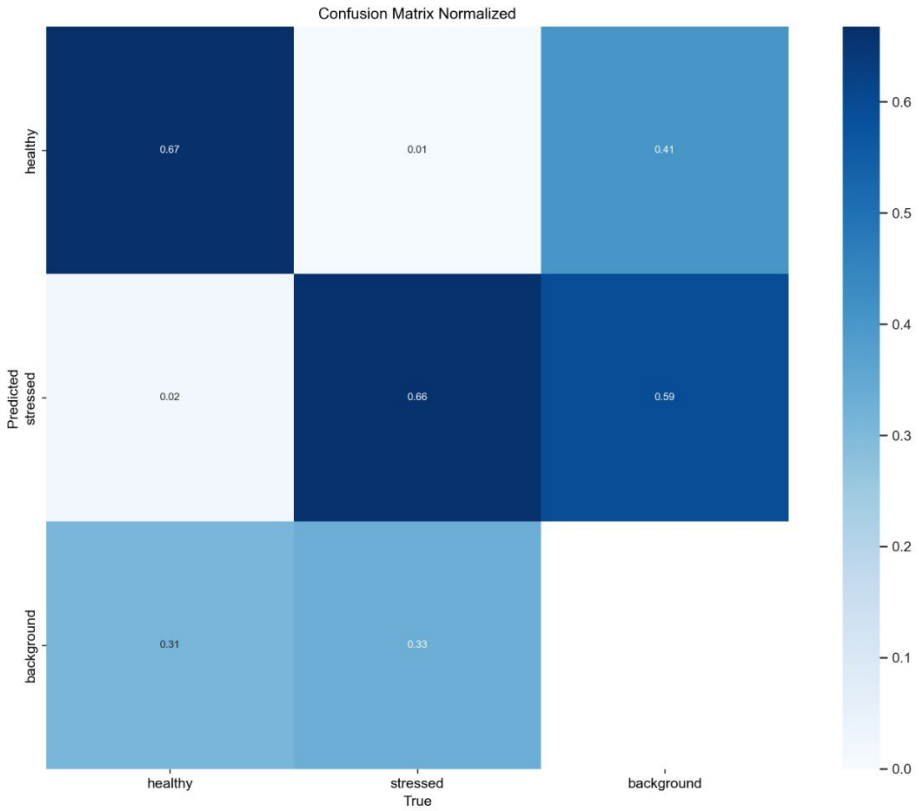


Figure 4.30: Confusion Matrix for RGBN images running 100 epochs using yolov8n

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	2067	0.82	0.744	0.82	0.533
Healthy	121	858	0.836	0.758	0.839	0.556
Stressed	121	1209	0.804	0.73	0.801	0.509
Epochs						150
Batch Size						16
Image Size						640
Model Size						yolov8n
Learning Rate						0.01

Table 4.19: YOLOv8n performance (150 epochs) on RGBN Images.

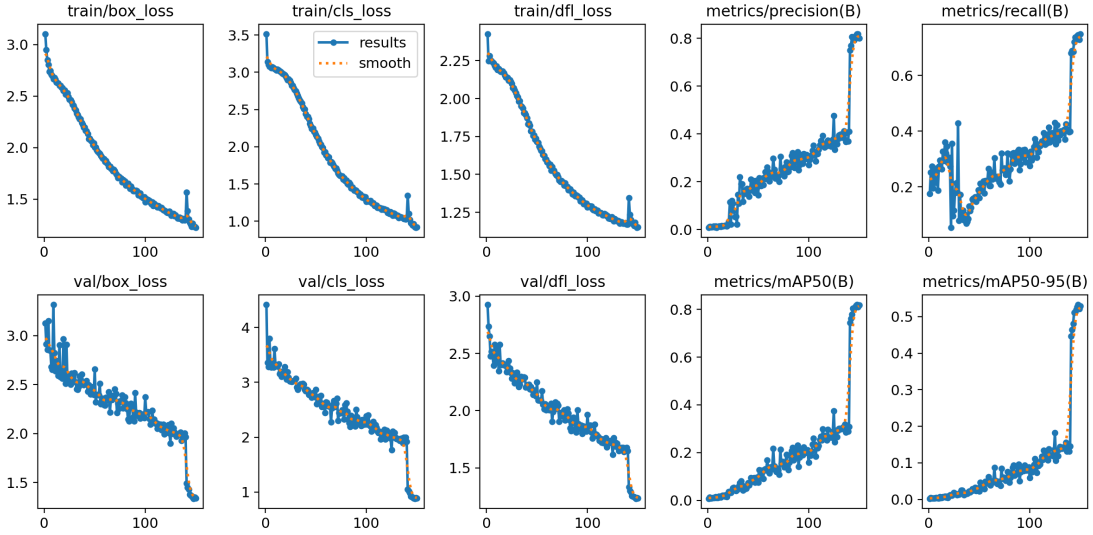


Figure 4.31: Training and Validation Loss and Metrics Plots

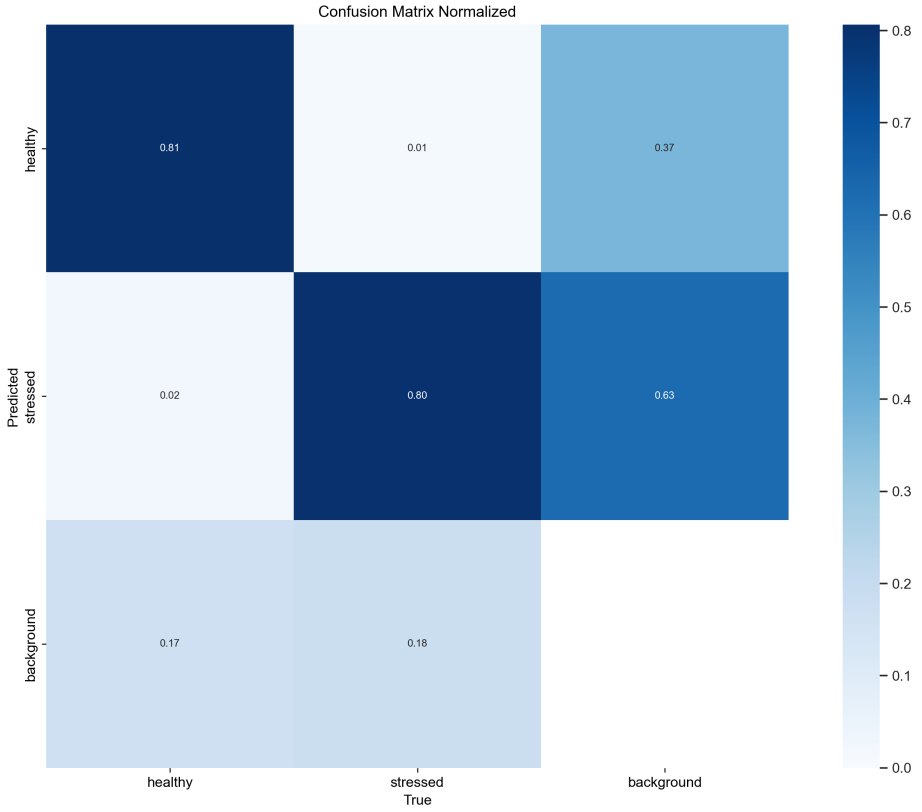


Figure 4.32: Confusion Matrix for RGBN images with 150 epochs

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	2025	0.894	0.86	0.925	0.65
Healthy	121	816	0.888	0.867	0.924	0.645
Stressed	121	1209	0.90	0.91	0.926	0.652
Epochs						80
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.20: YOLOv8s performance (80 epochs) on RGBN Images.

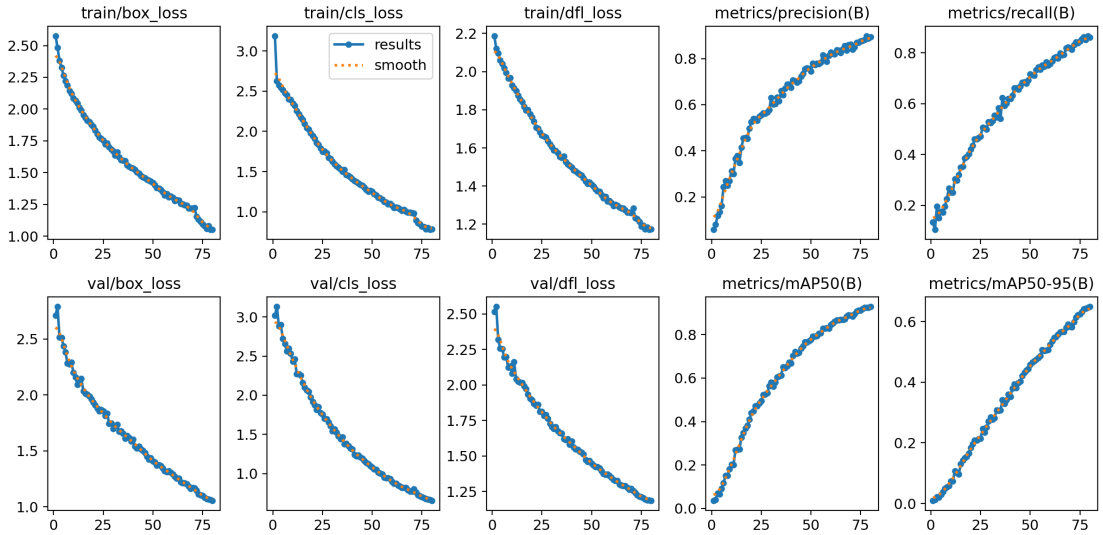


Figure 4.33: Training and Validation Loss and Metrics Plots

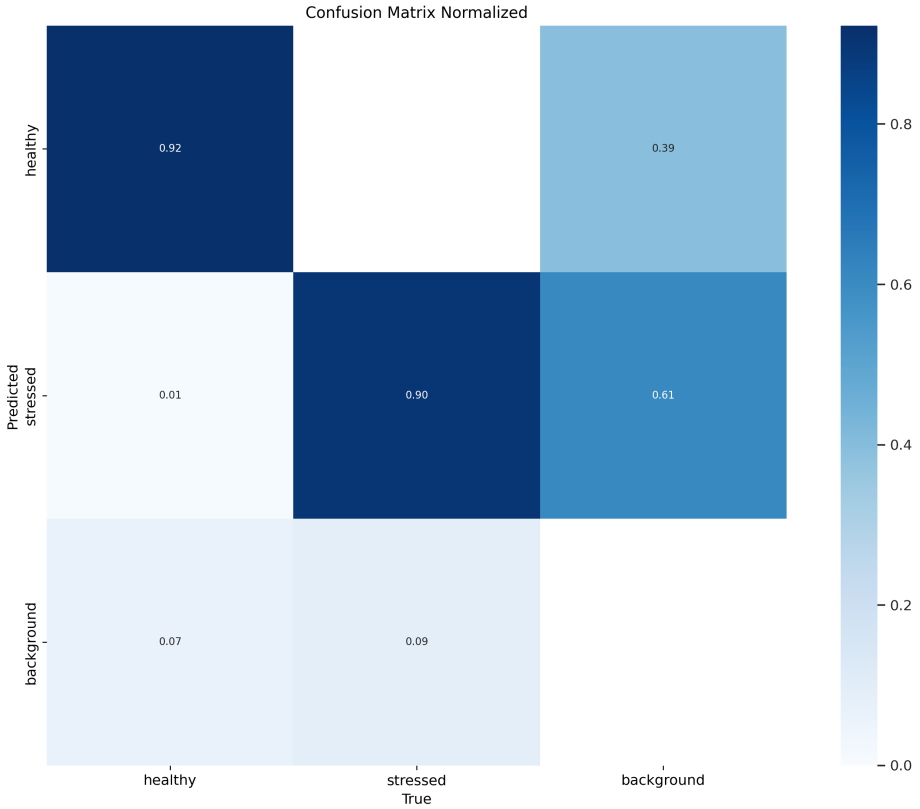


Figure 4.34: Confusion Matrix for RGBN images running 80 epochs using yolov8s

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	121	2025	0.921	0.911	0.96	0.724
Healthy	121	816	0.923	0.905	0.958	0.721
Stressed	121	1209	0.915	0.91	0.953	0.728
Epochs						100
Batch Size						32
Image Size						640
Model Size						yolov8s
Learning Rate						0.01

Table 4.21: YOLOv8s performance (100 epochs) on RGBN Images.

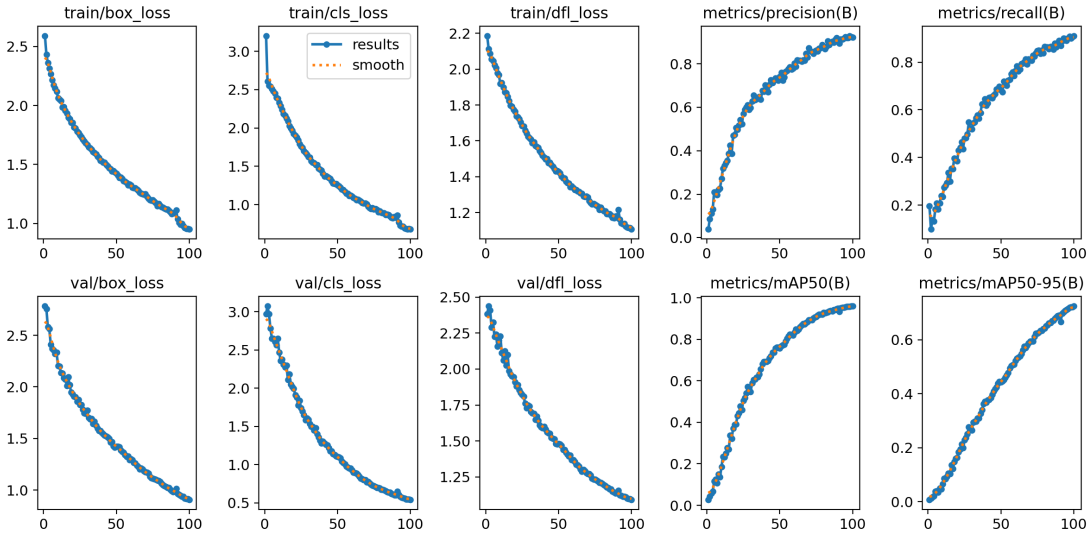


Figure 4.35: Training and Validation Loss and Metrics Plots

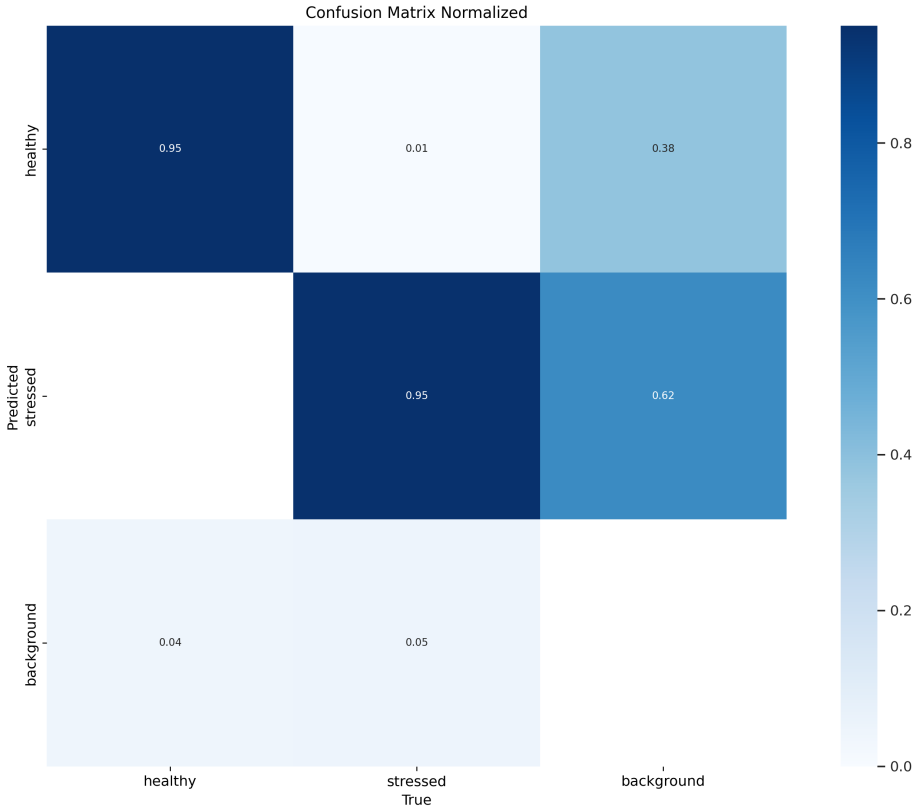


Figure 4.36: Confusion Matrix for RGBN images running 100 epochs using yolov8s

4.2.3 Faster R-CNN

Faster R-CNN was applied to both RGN and RGE multispectral image combinations to evaluate its performance in detecting stressed regions within the dataset. Two training configurations were tested: one with 50 epochs and another with 100 epochs. The primary objective was to determine the number of epochs required for the model to achieve stable results without overfitting.

RGN

Class	Images	Precision (P)	Recall (R)
All	121	0.8226	0.9018
Healthy	121	0.8231	0.9203
Stressed	121	0.8215	0.9015
Epochs			100
IoU			0.5
Batch Size			16
Image Size			640
Model Size			Faster R-CNN
Learning Rate			0.01

Table 4.22: Faster R-CNN performance (100 epochs) on RGN Images.

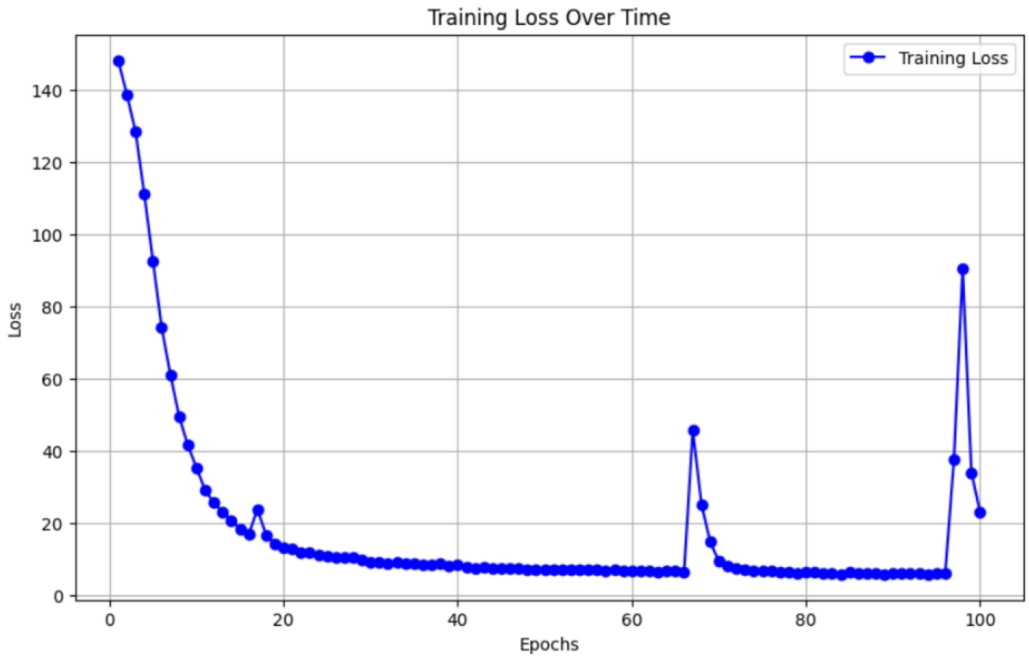


Figure 4.37: Training Plot

Training Loss and Model Behavior Analysis

Initial Rapid Learning Phase:
The steep decrease in training loss during the first 20 epochs suggests that the model is quickly learning the fundamental features of the data. After this initial

phase, the loss curve begins to flatten, indicating that the model is achieving a baseline level of performance.

Interpretation of Loss Stability:
The loss chart suggests that the model could have ceased improving around 50 epochs, where the training loss became relatively stable. However, since the validation/test performance continues to improve up to 100 epochs, this indicates that while the model's training loss may be plateauing, its ability to generalize is still being refined. A stable training loss plateau without further improvement would generally suggest the end of learning on the training set, but the observed enhancement in test/validation results suggests ongoing improvement in feature extraction on unseen data with further training.

Adjusting Learning Rate Based on Observations:
Based on these observations, the model was tested with 50 epochs using a lower learning rate of 0.001. The goal was to see if a lower learning rate could yield similar or improved generalization results within a shorter training period. This approach aimed to refine the model's learning trajectory by leveraging the benefits of an extended training session with a higher learning rate, and then focusing on convergence with a more conservative learning rate.

Class	**Images**	**Precision (P)**	**Recall (R)**
All	121	0.7599	0.8101
Healthy	121	0.7593	0.8127
Stressed	121	0.7584	0.8098
Epochs			50
IoU			0.5
Batch Size			16
Image Size			640
Model Size			Faster R-CNN
Learning Rate			0.001

Table 4.23: Faster R-CNN performance (50 epochs) on RGN Images.

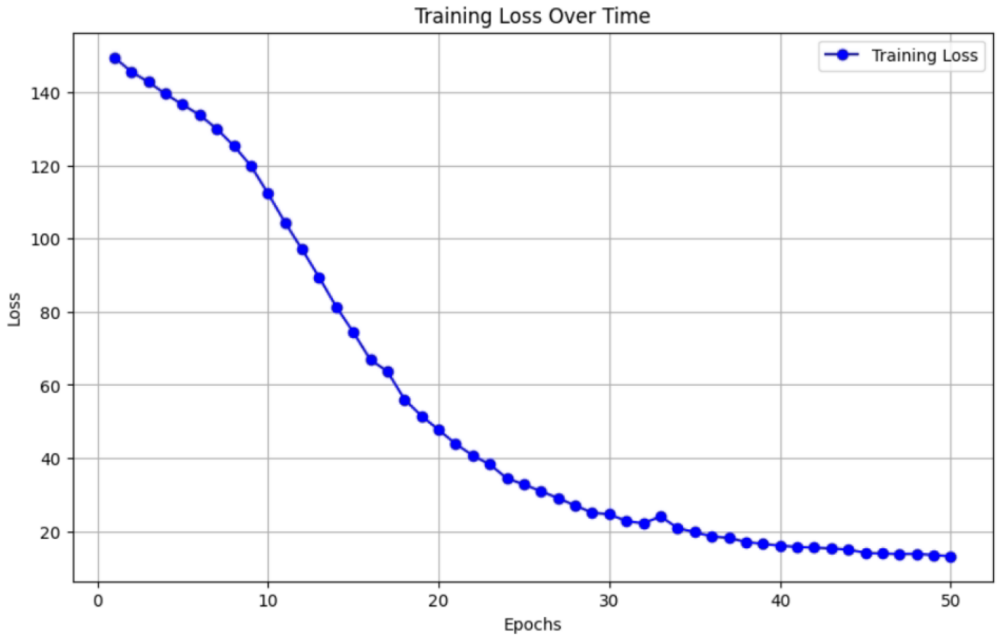


Figure 4.38: Training Plot

The loss chart for the model trained with 50 epochs shows a relatively stable and smooth curve, suggesting that the model has reached a point where it is consistently learning without fluctuations in training loss. In contrast, the model trained with 100 epochs exhibits some spikes in the later epochs, which could suggest moments of more aggressive learning, likely influenced by the higher learning rate (0.01 compared to 0.001 for 50 epochs).

While the 50-epoch model demonstrates stability, the 100-epoch model achieves better overall results, as indicated by improved test performance. This is likely because the additional training epochs allowed the model to continue refining its generalization capabilities, extracting features from the data more effectively despite occasional spikes in the training loss. Therefore, while the smoothness of the 50-epoch curve suggests consistent learning, the enhanced results with 100 epochs suggest that the model benefits from extended training and a higher learning rate to explore a broader feature space, ultimately improving its performance on unseen data.

The extended training duration with 100 epochs allows the model to refine its feature extraction process beyond what was achieved with the 50-epoch training session. Although a smooth curve, as seen with the 50-epoch training, generally indicates stable learning, the improved results observed with 100 epochs suggest that the model continues to benefit from additional epochs for further generalization. The decision to use a lower learning rate of 0.001 for 50 epochs was based on these observations, with the intent to explore if similar generalization could be achieved more quickly with a conservative learning approach. This strategy aligns

with the objective of balancing training efficiency and performance, maximizing generalization while minimizing overfitting.

RGE

Class	Images	Precision (P)	Recall (R)
All	121	0.9005	0.9547
Healthy	121	0.9132	0.9546
Stressed	121	0.891	0.9565
Epochs			50
IoU			0.5
Batch Size			16
Image Size			640
Model Size			Faster R-CNN
Learning Rate			0.001

Table 4.24: Faster R-CNN performance (50 epochs) on RGN Images.

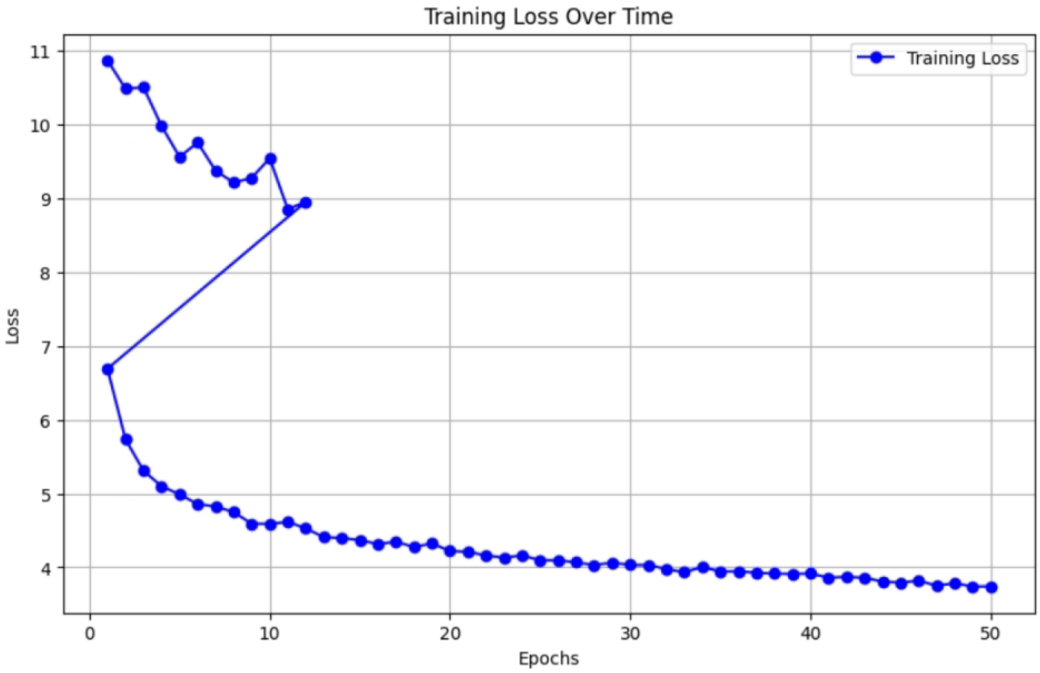


Figure 4.39: Training Plot

4.3 Discussion

In examining the impact of different hyperparameters on the performance of the YOLOv8n model, it was observed that adjustments to image size, batch size, and learning rate had significant effects on metrics such as precision, recall, and mean Average Precision (mAP). Table 4.25 explains the difference between these adjustments.

Hyperparameter	**Performance**	**Insight**
Image Size: 416 vs. 640	Larger image size (640) generally leads to improved precision and recall metrics due to the higher resolution, capturing more detailed features of the plants.	Increasing the image size enhances the model's ability to detect smaller or more detailed features, making it beneficial for datasets where fine-grained details contribute to classification.
Batch Size: 16 vs. 32	Batch size 32 shows slight improvements in recall and mAP metrics compared to batch size 16, indicating stability and minimal degradation with larger batches.	Suggests that YOLOv8n can handle larger batch sizes effectively, possibly reducing training time without significant loss in performance, making it suitable for applications with higher computational resources.
Learning Rate: 0.01, 0.1, 0.001	Learning rate 0.01 provides a balanced performance, while 0.1 may cause instability, and 0.001 could lead to slower convergence.	Optimal learning rates are essential for stable training. A learning rate of 0.01 achieves a good balance between convergence speed and stability, while higher values risk overshooting minima and lower values can result in prolonged training.

Table 4.25: YOLOv8n Hyperparameter Performance Insights

4.3.1 Comparative Analysis of Model Performance on RGE, RGN and RGB Images

Based on the results obtained from the experiments with YOLOv8n and YOLOv8s on both RGB and RGN datasets, the following observations were made:

It was observed that the YOLOv8n model on RGB images achieved a slightly higher mAP (50-95) across classes compared to RGN images, indicating better overall performance in fine-grained detection tasks. However, in terms of precision,

RGN images showed competitive values across classes, suggesting that the model effectively utilized the additional spectral band (NIR) in RGN. This highlights that YOLOv8n, even as a less complex model, was able to capture essential features from the multispectral data, with some metrics, such as recall, being slightly lower.

For yolov8s, models trained with RGB data, overall precision tends to be slightly higher across all classes compared to RGN. However, when adding the red-edge band (RGE), there is an improvement in precision for specific configurations, particularly evident in the mAP50 metric for RGE, where precision improves across both YOLOv8n and YOLOv8s models.

RGB generally shows better recall across classes, which suggests a broader capability in capturing the full range of instances without missing detections. This is consistent across models like YOLOv8n and YOLOv8s. The RGN configuration slightly improves recall in detecting stressed classes, implying it might be more sensitive to features indicative of plant stress, likely due to the inclusion of the near-infrared band which is sensitive to vegetation health.

While RGB tends to maintain higher mAP50-95 scores, indicating better performance across various IoU thresholds, RGE configurations show a substantial increase in mAP50 compared to RGB, particularly in fine-grained detection tasks where boundary precision is essential. For RGN, we observe improvements in mAP50, which enhances the model's capability in more localized detections. However, RGB generally retains a better mAP50-95 score, suggesting it captures a broader range of features effectively.

With pretrained Hugging Face models, we see improved precision and recall on both RGN and RGE configurations. This suggests that using transfer learning allows the model to leverage already-learned features, enhancing its ability to generalize and capture fine details specific to plant leaf classification tasks. The pretrained models on RGE consistently outperform in both mAP50 and mAP50-95, providing more robust detection capabilities with enhanced precision and recall, especially in distinguishing between healthy and stressed plants.

Faster R-CNN models show a noticeable improvement when trained on multispectral data (RGE and RGN) over RGB. This model appears to benefit from the additional spectral bands, showing higher precision and recall values in stressed crop detection, which suggests a better adaptation to complex scenes and fine-tuning of boundary-level distinctions. Both RGE and RGN achieve comparable performance to RGB on Faster R-CNN but tend to outperform on specialized metrics, particularly in precision and recall for the stressed class, indicating an enhanced capability in detecting subtle stress indicators in plants.

In conclusion, RGB generally performs well across broader metrics, but RGE and RGN configurations reveal specific advantages for plant stress detection. The addition of red-edge and near-infrared bands in RGE and RGN allows the models to capture vegetation health indicators more effectively, as evidenced by the

higher mAP50 in RGE configurations and improved recall in stressed classes for RGN. Pretrained models on RGE datasets consistently demonstrate the highest performance, suggesting that transfer learning combined with multispectral data enhances detection accuracy. Faster R-CNN further supports the notion that more complex architectures benefit from multispectral data, achieving high precision and recall in plant-specific applications.

4.3.2 Comparative Analysis of Model Performance on RGREN, RGBN and RGB Images

For All classes, RGBN shows improved precision over RGB. RGB achieves a precision of around 0.8233, whereas RGBN configuration reaches approximately 0.836. This increase suggests that the addition of the NDVI channel helps the model better differentiate between object classes overall. Healthy class benefits notably with the RGBN setup, reaching precision values close to 0.923 in YOLOv8m, surpassing RGB's performance of around 0.895. This improvement implies that the extra spectral information aids in the classification accuracy of healthy instances. In the Stressed class, RGREN maintains a competitive precision, although it's slightly below RGBN in YOLOv8m. However, it still indicates a robust performance, which highlights the utility of red-edge and NDVI data in detecting stressed plants effectively.

The recall for All classes is slightly higher for the RGB configuration compared to RGREN, but both RGBN and RGREN configurations surpass RGB for individual class recall rates in certain conditions. In Healthy and Stressed classes, RGBN shows improved recall in YOLOv8m compared to RGB and YOLOv8n. This suggests that the additional channels, particularly NDVI, assist in more effectively capturing stressed vegetation patterns.

For All classes, both RGBN and RGREN improve upon RGB in mAP50. Specifically, RGBN reaches an mAP50 of around 0.839 in YOLOv8n, whereas RGB lags behind with a maximum mAP50 of 0.8071 for YOLOv8n configurations. This increase implies that the model benefits in localization and recognition with the added spectral information. Healthy and Stressed classes show notable improvements in mAP50 with the RGREN configuration, where mAP50 can reach around 0.953 in YOLOv8m, compared to RGB's 0.931 in YOLOv8s for stressed classes. This enhanced performance demonstrates the added value of the red-edge channel in detecting more subtle features in plant health status.

For All classes, RGBN performs slightly better than RGB in mAP50-95, indicating improved detection across various IoU thresholds. RGB achieves a maximum of approximately 0.6614, while RGBN can achieve up to 0.728 with YOLOv8m. This metric improvement reflects that RGBN generalizes better across different object sizes and positions. In the Healthy and Stressed classes, RGBN configurations

maintain a competitive mAP50-95, again underscoring the potential advantage of multispectral data for specific vegetation detection tasks.

The results indicate that using additional spectral bands (NDVI and red-edge in RGBN and RGREN configurations) consistently enhances the model's ability to detect and classify instances with higher precision and mAP metrics compared to RGB alone. The improvements in the Healthy and Stressed classes suggest that spectral data provides significant benefits for distinguishing stressed crops. This evidence supports the integration of multispectral data for more complex object detection tasks in precision agriculture, where subtle differences in plant health need to be accurately captured.

4.3.3 Comparative Analysis of Model Performance with State of the Art

In addition to evaluating the effects of hyperparameter adjustments, the performance of the YOLOv8n and YOLOv8s models was benchmarked against existing state-of-the-art methods for plant health assessment. This comparison included methods such as Retina-UNet-Ag, Mask R-CNN, and RetinaNet. The models were evaluated separately for the Healthy and Stressed classes, providing a comprehensive understanding of performance in detecting different plant health statuses.

The following tables present a comparative analysis between the YOLOv8n and YOLOv8s models and the selected state-of-the-art methods, focusing on metrics such as Intersection over Union (IoU), Precision, and Recall for each class.

Model	**IoU**	**Precision**	**Recall**
Retina-UNet-Ag	0.574	0.659	0.832
Mask R-CNN	0.556	0.644	0.769
RetinaNet	0.537	0.578	0.899
Faster R-CNN (state-of-the-art)	0.563	0.630	0.891
YOLO v3	0.487	0.541	0.855
YOLOv8n (thesis)	0.7	0.827	0.81
YOLOv8m (thesis)	0.7	0.943	0.892

Table 4.26: Comparison of YOLOv8m, YOLOv8n with state-of-the-art methods for Healthy class in RGB Images.

Model	IoU	Precision	Recall
Retina-UNet-Ag	0.604	0.702	0.841
Mask R-CNN	0.598	0.700	0.809
RetinaNet	0.583	0.698	0.795
Faster R-CNN (state-of-the-art)	0.554	0.781	0.654
YOLO v3	0.394	0.407	0.882
YOLOv8n (thesis)	0.7	0.797	0.747
YOLOv8m (thesis)	0.7	0.939	0.858

Table 4.27: Comparison of YOLOv8m, YOLOv8n, and Faster R-CNN (thesis) with state-of-the-art methods for Stressed class.

The performance of YOLOv8n and YOLOv8m (as used in this thesis) on RGB images compares favorably against state-of-the-art models like Faster R-CNN, RetinaNet, and Mask R-CNN. YOLOv8n achieves a significantly higher precision (0.827) and comparable recall (0.81) in the Healthy class, surpassing Faster R-CNN in precision. Similarly, for the Stressed class, YOLOv8m attains a high precision of 0.939, outperforming the state-of-the-art models in both precision and recall, illustrating the efficacy of the YOLOv8 architecture on RGB images in terms of accuracy and object detection performance.

Model	IoU	Precision	Recall
Retina-UNet-Ag	0.381	0.497	0.677
YOLOv8n (Thesis)	0.7	0.778	0.674
YOLOv8m (Thesis)	0.7	0.908	0.87
Faster R-CNN (Thesis)	0.5	0.823	0.920
Pretrained Hugging Face (Thesis)	0.7	0.972	0.959

Table 4.28: Performance metrics for RGN - Healthy class

Model	IoU	Precision	Recall
Retina-UNet-Ag	0.419	0.488	0.752
YOLOv8n (Thesis)	0.7	0.793	0.706
YOLOv8m (Thesis)	0.7	0.902	0.802
Faster R-CNN (Thesis)	0.5	0.821	0.901
Pretrained Hugging Face (Thesis)	0.7	0.969	0.958

Table 4.29: Performance metrics for RGN - Stressed class

For RGN images, the models trained in this thesis exhibit superior performance compared to state-of-the-art methods, particularly in terms of precision. In the Healthy class, the Pretrained Hugging Face model achieves the highest precision

of 0.972 and recall of 0.959, significantly outperforming Retina-UNet-Ag, which only reached a precision of 0.497 and recall of 0.677. Similarly, for the Stressed class, the Pretrained Hugging Face model shows a remarkable improvement with precision and recall values of 0.969 and 0.958, respectively. This demonstrates that the RGN-based models, particularly with the pretrained architecture, provide more accurate and reliable detection capabilities for both healthy and stressed instances, highlighting the advantage of multispectral data over traditional state-of-the-art models when applied to complex agricultural tasks.

Model	**IoU**	**Precision**	**Recall**
Retina-UNet-Ag	0.380	0.466	0.669
YOLOv8n (Thesis)	0.7	0.658	0.483
YOLOv8m (Thesis)	0.7	0.913	0.900
Faster R-CNN (Thesis)	0.5	0.880	0.954
Pretrained Hugging Face (Thesis)	0.7	0.959	0.964

Table 4.30: Performance metrics for RGE - Healthy class

Model	**IoU**	**Precision**	**Recall**
Retina-UNet-Ag	0.398	0.462	0.763
YOLOv8n (Thesis)	0.7	0.627	0.500
YOLOv8m (Thesis)	0.7	0.900	0.910
Faster R-CNN (Thesis)	0.5	0.891	0.956
Pretrained Hugging Face (Thesis)	0.7	0.959	0.961

Table 4.31: Performance metrics for RGE - Stressed class

In comparing the state-of-the-art models with the RGE configurations, it is evident that utilizing the multispectral RGE data improves detection metrics, particularly in precision and recall for certain models. For the Healthy class, the pretrained Hugging Face model achieves the highest precision and recall, with values of 0.959 and 0.964, respectively, outperforming models like Retina-UNet-Ag, which shows significantly lower precision at 0.466. Similarly, for the Stressed class, the pretrained Hugging Face model attains high precision and recall values (0.959 and 0.959), demonstrating enhanced performance over models like YOLOv8n, which achieves lower recall at 0.627. This illustrates that using RGE data with more advanced models can capture detailed features and improve generalization capabilities, especially when distinguishing between healthy and stressed vegetation.

Configuration	Model	Precision	Recall	mAP50	mAP50-95
RGBN	YOLOv8n	0.836	0.758	0.839	0.556
	YOLOv8m	0.923	0.905	0.958	0.721
RGREN	YOLOv8n	0.813	0.695	0.781	0.512
	YOLOv8m	0.854	0.693	0.818	0.514

Table 4.32: Comparison of YOLOv8 Performance on RGB+NDVI and RGREN Configurations for Healthy class

Configuration	Model	Precision	Recall	mAP50	mAP50-95
RGBN	YOLOv8n	0.804	0.73	0.801	0.509
	YOLOv8m	0.915	0.91	0.953	0.728
RGREN	YOLOv8n	0.834	0.696	0.783	0.498
	YOLOv8m	0.847	0.694	0.808	0.501

Table 4.33: Comparison of YOLOv8 Performance on RGBN and RGREN Configurations for Stressed class

In comparing the RGB+NDVI and RGREN configurations, it becomes clear that the addition of NDVI (RGBN) offers improved performance metrics, particularly in terms of recall and mAP scores for the YOLOv8 models. For the Healthy class, the YOLOv8m model under RGB+NDVI achieves higher recall and mAP50 values (0.905 and 0.958, respectively), significantly outperforming the RGREN configuration, which records a recall of 0.693 and an mAP50 of 0.818. Additionally, in the Stressed class, YOLOv8m with RGB+NDVI again demonstrates superior performance, attaining a recall of 0.91 and an mAP50-95 of 0.728, compared to RGREN's recall of 0.694 and mAP50-95 of 0.501 with the same model. These results highlight that while both RGB+NDVI and RGREN leverage additional spectral bands, RGB+NDVI consistently provides enhanced detection and classification capabilities across both healthy and stressed vegetation instances, especially when applied to more refined models like YOLOv8m.

Chapter 5

Conclusion

The research conducted in this thesis aimed to improve the accuracy and robustness of water stress detection in potato crops through the application of advanced deep learning models, specifically YOLOv8 and Faster R-CNN, on RGB and multispectral imaging datasets, including RGB-NIR and RGB-Red-Edge combinations.

The primary findings indicate significant improvements in detection capabilities when incorporating multispectral data. The inclusion of additional bands, such as NIR and Red-Edge, proved beneficial in distinguishing between healthy and stressed vegetation. This enhancement is particularly evident in the higher recall scores for stressed plant detection achieved with RGB+NDVI and RGE configurations using YOLOv8s. Additionally, Faster R-CNN has shown enhanced recall with RGN and RGE configurations. Notably, the pretrained Hugging Face model on RGE and RGN data yielded exceptional precision and recall values, which underscores the advantage of using specific spectral bands for detecting subtle signs of stress. Similarly, YOLOv8 models with RGB+NDVI data achieved strong mAP50, mAP50-95 scores, reflecting the effectiveness of NDVI in capturing nuanced crop health details.

Furthermore, the integration of Red-Edge bands in the RGE configuration provided a moderate improvement in edge-sensitive tasks, highlighting the value of multispectral imaging in agricultural applications where boundary delineation is critical. These findings support the utility of specific multispectral indices in detecting variations in crop health that are not always visible in the standard RGB spectrum. By enabling more accurate stress detection, the methods explored in this research contribute to the goals of precision agriculture, promoting timely interventions and potentially increasing crop yield.

In conclusion, this thesis confirms that multispectral imaging, combined with advanced deep learning architectures, improves the precision and generalization capabilities of stress detection systems in potato crops. The research establishes that integrating additional spectral data with RGB not only enhances the model's

robustness but also mitigates the impact of environmental variability.

Future work could involve further enhancing image resolution to capture finer details of crop health. Additionally, incorporating other indices, such as the Green Normalized Difference Vegetation Index (GNDVI), and exploring alternative deep learning models could provide further gains in detection accuracy. These advancements would contribute to the scalability and applicability of automated crop stress management systems across diverse agricultural settings, aligning with the principles of sustainable farming.

Appendix A

Qualitative Results

A.1 Segmentation and NDVI Results

In this study, segmentation and NDVI (Normalized Difference Vegetation Index) were initially explored to detect water-stressed plants. The segmentation approach combined color-based and threshold-based methods to identify plants within the image, followed by calculating NDVI to assess plant health.

The NDVI was calculated at three different levels:

1. For the entire image, covering all plants at once.
2. For each individual leaf in the segmented regions.
3. For each plant, by aggregating NDVI values from the segmented leaves.

NDVI for the Entire Image

In this approach, NDVI was calculated for the entire image, encompassing all plants within the frame. This provided a general assessment of the health status across the entire field or region of interest. Figure A.1 shows the results of this whole-image NDVI calculation, where higher NDVI values indicate healthier vegetation, and lower values may suggest stressed or unhealthy plants.

Plant Detection Using Threshold on NDVI

A threshold was applied to the calculated NDVI values in the image to detect and segment individual plants based on their health status. Plants with higher NDVI values were detected and highlighted, as shown in Figure A.2. This method allowed for a clear distinction between stressed and healthy plants by segmenting areas where NDVI values met or exceeded a certain threshold.

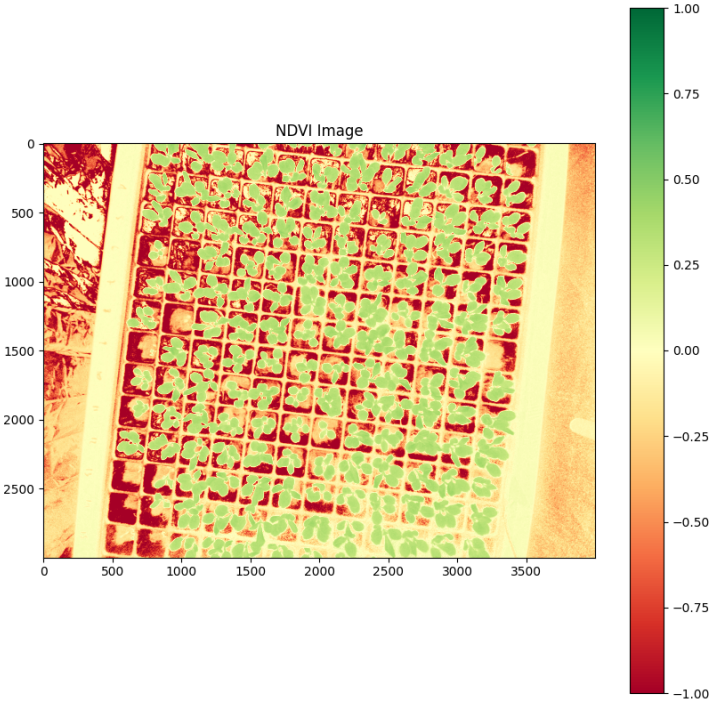


Figure A.1: NDVI calculated for the entire image showing the overall plant health within the field.

NDVI for Each Leaf

To provide a more granular understanding of plant health, NDVI was calculated for each leaf within the segmented regions of the plants. This approach enabled the detection of stressed regions within individual plants, which could be critical for early intervention. Figure A.3 illustrates the NDVI values calculated for individual leaves, highlighting variations in health across different parts of the plant.

NDVI for Each Plant

Finally, the NDVI values were aggregated for each plant by calculating the NDVI for individual leaves and then averaging these values to estimate the overall health of each plant. Figure A.4 demonstrates how this method provides a plant-level assessment of health, which is crucial for precision agriculture and targeted interventions.

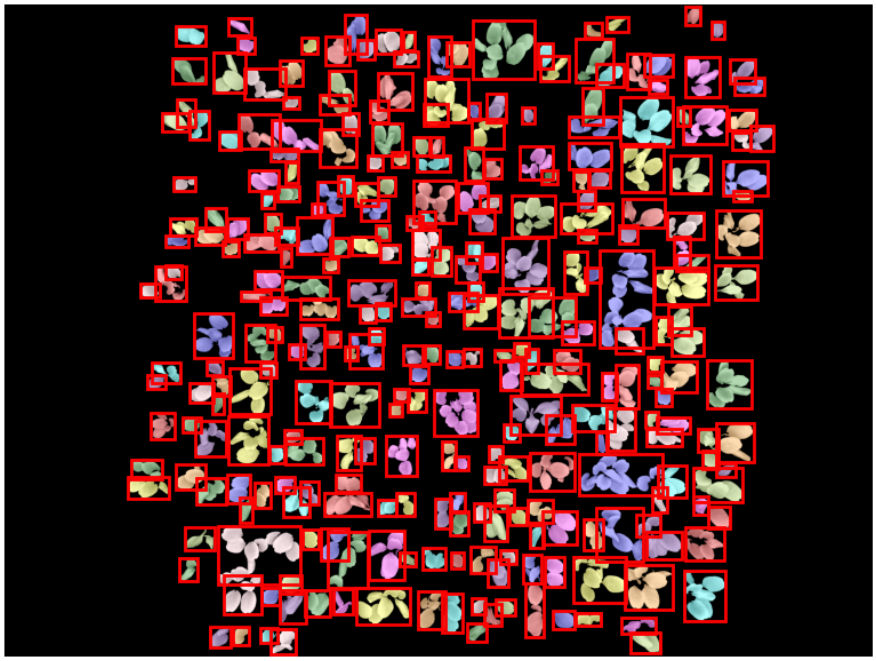


Figure A.2: Detection of individual plants using a threshold applied to the NDVI values. Higher NDVI values indicate healthy plants.

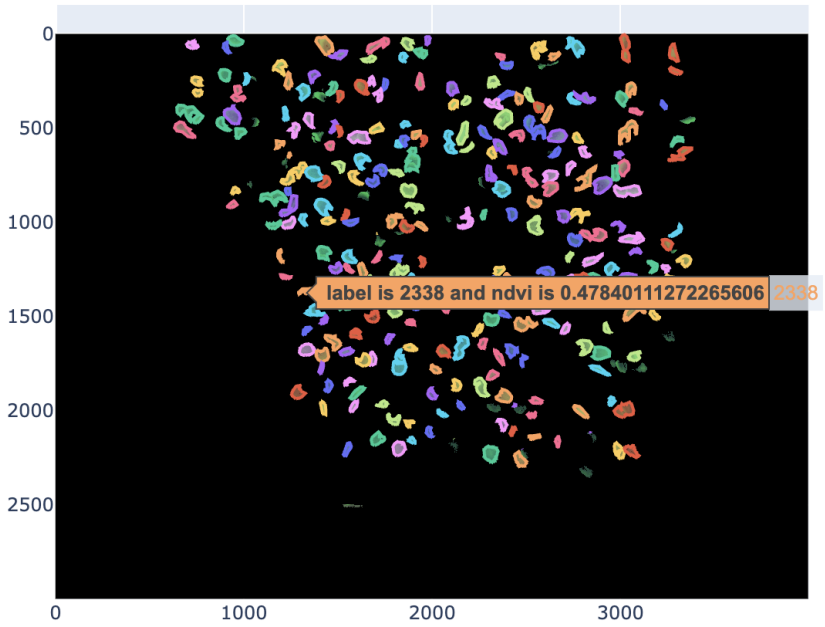


Figure A.3: NDVI calculated for each leaf within the segmented regions of the plants, showing health variations across leaves.

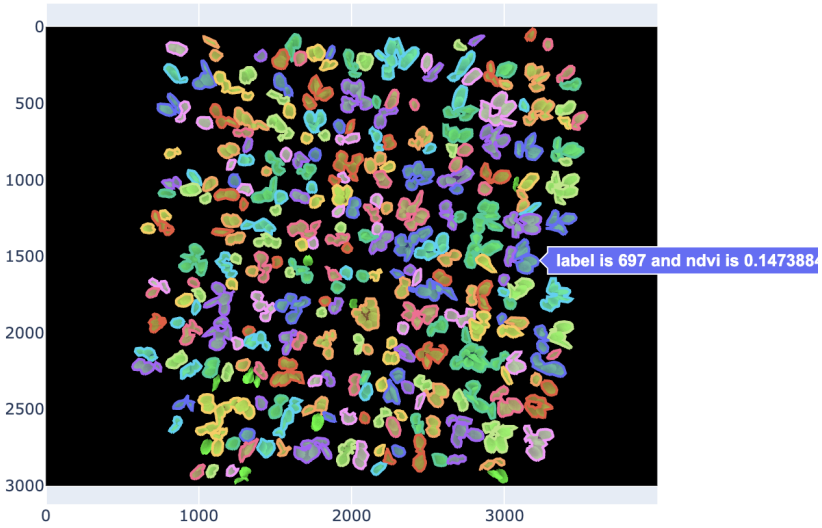


Figure A.4: NDVI calculated for each plant by aggregating NDVI values from individual leaves. This provides an overall health assessment of the plant.

These qualitative results demonstrate the potential of integrating image segmentation with NDVI to assess plant health. Although quantitative evaluation such as precision, recall, and IoU was not performed, the segmentation approach and NDVI calculations provide valuable insights into detecting water-stressed areas. Further evaluation would be needed to fully validate these methods in a real-world agricultural setting.

Bibliography

[1] Xing Yang, Lei Shu, Jianing Chen, Mohamed Amine Ferrag, Jun Wu, Edmond Nurellari, and Kai Huang. «A Survey on Smart Agriculture: Development Modes, Technologies, and Security and Privacy Challenges». In: *IEEE/CAA Journal of Automatica Sinica* 8.2 (2021), pp. 273–302. DOI: 10.1109/JAS.2020.1003536 (cit. on p. 1).

[2] E.s Mohamed, Abdelaziz Belal, Sameh Kotb Abd-Elmabod, Mohammed El-Shirbeny, Abd-Alla Gad, and Mohamed Zahran. «Smart farming for improving agricultural management». In: *The Egyptian Journal of Remote Sensing and Space Science* 24 (Sept. 2021). DOI: 10.1016/j.ejrs.2021.08.007 (cit. on pp. 1, 2).

[3] George Adamides et al. «Smart farming techniques for climate change adaptation in Cyprus». In: *Atmosphere* 11.6 (2020), p. 557 (cit. on p. 1).

[4] Bellvert, J., Zarco-Tejada, P.J., Girona, and J. et al. «Mapping crop water stress index in a 'Pinot-noir' vineyard: comparing ground measurements with thermal remote sensing imagery from an unmanned aerial vehicle». In: *Precision Agriculture* 15.4 (2014), pp. 361–376. DOI: 10.1007/s11119-013-9334-5 (cit. on pp. 2, 5).

[5] Food and Agriculture Organization of the United Nations. *The Future of Food and Agriculture: Trends and Challenges*. Accessed: 2024-09-12. Food and Agriculture Organization of the United Nations, 2018. URL: https://www.fao.org/3/i9553en/I9553EN.pdf (cit. on pp. 2, 5).

[6] X. Li, Y. Zhang, Y. Liu, and Y. He. «Applications of machine learning and IoT in precision agriculture: A review». In: *Sensors* 21.5 (2021), p. 1717. DOI: 10.3390/s21051717. URL: https://www.mdpi.com/1424-8220/21/5/1717 (cit. on pp. 2, 9, 11).

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. «You Only Look Once: Unified, Real-Time Object Detection». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91 (cit. on p. 3).

[8] S. Ren, K. He, R. Girshick, and J. Sun. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». In: *Advances in Neural Information Processing Systems*. Vol. 28. 2015, pp. 91–99. DOI: 10.1109/CVPR.2015.7298953 (cit. on pp. 3, 21).

[9] O. Ronneberger, P. Fischer, and T. Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». In: *Medical Image Computing and Computer-Assisted Intervention*. 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28 (cit. on p. 3).

[10] K. Heand X. Zhang, S. Ren, and J. Sun. «Deep Residual Learning for Image Recognition». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on p. 3).

[11] Andreas Kamilaris and Francesc Xavier Prenafeta-Boldú. «Deep learning in agriculture: A survey». In: *Computers and Electronics in Agriculture* 147 (2018), pp. 70–90. DOI: 10.1016/j.compag.2018.02.016. URL: https://doi.org/10.1016/j.compag.2018.02.016 (cit. on pp. 3, 5, 6, 12, 24).

[12] A. Dosovitskiy et al. «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale». In: *arXiv preprint arXiv:2010.11929* (2020). URL: https://arxiv.org/abs/2010.11929 (cit. on p. 3).

[13] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah. «Transformers in Vision: A Survey». In: *arXiv preprint arXiv:2101.01169* (2021). URL: https://arxiv.org/abs/2101.01169 (cit. on p. 3).

[14] J. Geipel, J. Link, and W. Claupein. «Combined Spectral and Spatial Modeling Improves the Remote Estimation of Nitrogen Status in Wheat Canopies». In: *Remote Sensing* 6.11 (2014), pp. 10648–10664. DOI: 10.3390/rs61110648 (cit. on p. 3).

[15] J. G. A. Barbedo. «A Review on the Use of Unmanned Aerial Vehicles and Imaging Sensors for Monitoring and Assessing Plant Stresses». In: *Drones* 3.2 (2019), p. 40. DOI: 10.3390/drones3020040 (cit. on p. 3).

[16] L. Zheng, J. Zhou, X. Song, and Y. Liang. «IoT Applications in Precision Agriculture: A Review». In: *IEEE Internet of Things Journal* 9.14 (2022), pp. 11730–11743. DOI: 10.1109/JIOT.2021.3116745 (cit. on pp. 3, 4, 10).

[17] J. Jin, Q. Wang, and D. Wang. «Smart Irrigation with Wireless Sensors: Performance Evaluation and Optimization». In: *Computers and Electronics in Agriculture* 169 (2020), p. 105227. DOI: 10.1016/j.compag.2019.105227 (cit. on pp. 3, 10, 11).

[18] R. Ferguson and D. Wu. «IoT Applications in Agriculture: Growing the Future of Farming». In: *IEEE Internet of Things Journal* 7.12 (2020), pp. 11719–11725. DOI: 10.1109/JIOT.2020.2994121 (cit. on pp. 3, 11).

[19] L. Wang, W. Hu, Y. Liu, and Y. Zhang. «Plant-Based Sensor Technologies for Smart Agriculture: A Review». In: *Biosensors & Bioelectronics* 167 (2020), p. 112432. DOI: 10.1016/j.bios.2020.112432 (cit. on pp. 3, 9).

[20] X. Long, Z. He, Y. Liu, and Y. Xu. «A Smart Irrigation System with IoT Based Monitoring». In: *IEEE Internet of Things Journal* 6.2 (2019), pp. 1315–1326. DOI: 10.1109/JIOT.2018.2883031 (cit. on pp. 4, 10).

[21] J. Balendonck, A. L. Sanchez, and J. D. Carrillo. «Advances in Smart Irrigation Systems: IoT, Sensors, and Data Analysis». In: *Sensors Journal* 20.1 (2020), pp. 285–295. DOI: 10.3390/s20010285 (cit. on p. 4).

[22] A. Kaloxylos and et.al. «Agricultural Internet of Things and Decision Support Systems for Precision Smart Farming». In: *Biosystems Engineering* 153 (2017), pp. 123–131. DOI: 10.1016/j.biosystemseng.2017.05.016 (cit. on pp. 4, 10).

[23] Shamshiri R R, Kalantari F, Ting K C, Thorp K R, Hameed I A, Weltzien C, and et al. «Advances in greenhouse automation and controlled environment agriculture: A transition to plant factories and urban agriculture». In: *International Journal of Agricultural and Biological Engineering* 11.1 (2018), pp. 1–22. DOI: 10.25165/j.ijabe.20181101.3210 (cit. on p. 5).

[24] Konstantinos G. Liakos, Patrizia Busato, Dimitrios Moshou, Simon Pearson, and Dionysis Bochtis. «Machine Learning in Agriculture: A Review». In: *Sensors* 18.8 (2018). Submission received: 27 June 2018 / Revised: 31 July 2018 / Accepted: 7 August 2018 / Published: 14 August 2018, p. 2674. DOI: 10.3390/s18082674. URL: https://www.mdpi.com/1424-8220/18/8/2674 (cit. on pp. 5, 6).

[25] H. Charles J. Godfray et al. «Food Security: The Challenge of Feeding 9 Billion People». In: *Science* 327.5967 (2010), pp. 812–818. DOI: 10.1126/science.1185383 (cit. on pp. 5, 6).

[26] Xia Deng, Yan Li, and Yue Zhang. «An IoT-based smart irrigation system for agriculture». In: *IEEE Access* 8 (2020), pp. 54803–54812. DOI: 10.1109/ACCESS.2020.2981679 (cit. on p. 5).

[27] J. Torres-Sánchez, J.M. Peña, A.I. de Castro, and F. López-Granados. «Multi-temporal mapping of the vegetation fraction in early-season wheat fields using images from UAV». In: *Computers and Electronics in Agriculture* 103 (2014), pp. 104–113. DOI: 10.1016/j.compag.2014.02.009 (cit. on p. 5).

[28] T. Wheeler and J. von Braun. «Climate change impacts on global food security». In: *Science* 341.6145 (2013), pp. 508–513. DOI: 10.1126/science.1239402 (cit. on pp. 5, 6).

[29] Sonka Steven. «Big data and the ag sector: More than lots of numbers». In: *International Food and Agribusiness Management Review* 17.1 (2014), pp. 1–20. DOI: 10.22004/ag.econ.164988 (cit. on p. 6).

[30] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, and Prabhat. «Deep learning and process understanding for data-driven Earth system science». In: *Nature* 566 (2019), pp. 195–204. DOI: 10.1038/s41586-019-0912-1 (cit. on p. 6).

[31] J. Foley et al. «Solutions for a cultivated planet». In: *Nature* 478 (2011), pp. 337–342. DOI: 10.1038/nature10452. URL: https://doi.org/10.1038/nature10452 (cit. on p. 6).

[32] A. Matese et al. «Intercomparison of UAV, Aircraft and Satellite Remote Sensing Platforms for Precision Viticulture». In: *Remote Sensing* 7.3 (2015), pp. 2971–2990. DOI: 10.3390/rs70302971. URL: https://doi.org/10.3390/rs70302971 (cit. on pp. 6, 9).

[33] T. C. Hsiao. «Plant Responses to Water Stress». In: *Annual Review of Plant Physiology* 24 (1973), pp. 519–570 (cit. on p. 6).

[34] P. J. Kramer and J. S. Boyer. *Water Relations of Plants and Soils*. San Diego, CA: Academic Press, 1995 (cit. on p. 6).

[35] J. T. Ritchie. «Water Dynamics in the Soil-Plant-Atmosphere System». In: *Plant and Soil* 58.1 (1981), pp. 81–96 (cit. on p. 6).

[36] M. M. Chaves, J. P. Maroco, and J. S. Pereira. «Understanding Plant Responses to Drought—From Genes to the Whole Plant». In: *Functional Plant Biology* 30 (2003), pp. 239–264. DOI: 10.1071/FP02076 (cit. on p. 6).

[37] F. Tardieu. «Any Trait or Trait-Related Allele Can Confer Drought Tolerance: Just Design the Right Selection Environment». In: *Journal of Experimental Botany* 63.1 (2012), pp. 25–31. DOI: 10.1093/jxb/err269 (cit. on p. 6).

[38] C. B. Field et al., eds. *Climate Change 2014 – Impacts, Adaptation and Vulnerability: Part A: Global and Sectoral Aspects*. Cambridge, UK and New York, NY, USA: Cambridge University Press, 2014. ISBN: 9781107058071 (cit. on p. 6).

[39] E. Fereres and M. A. Soriano. «Deficit irrigation for reducing agricultural water use». In: *Journal of Experimental Botany* 58.2 (2006), pp. 147–159. ISSN: 0022-0957. DOI: 10.1093/jxb/erl165. URL: https://doi.org/10.1093/jxb/erl165 (cit. on pp. 6, 8).

[40] P. J. Zarco-Tejada, V. González-Dugo, and J. A. J. Berni. «Fluorescence, Temperature and Narrow-Band Indices Acquired from a UAV Platform to Detect Water Stress in Almond Trees». In: *Remote Sensing of Environment* 126 (2012), pp. 14–25. DOI: 10.1016/j.rse.2012.08.010 (cit. on pp. 6, 7).

[41] J. Baluja, M. P. Diago, P. Balda, R. Zorer, F. Meggio, F. Morales, and J. Tardaguila. «Assessment of Vineyard Water Status Variability by Thermal and Multispectral Imagery Using an Unmanned Aerial Vehicle (UAV)». In: *Irrigation Science* 30.6 (2012), pp. 511–522. DOI: 10.1007/s00271-012-0383-4 (cit. on p. 6).

[42] A. Chlingaryan, S. Sukkarieh, and B. Whelan. «Machine Learning Approaches for Crop Yield Prediction and Nitrogen Status Estimation in Precision Agriculture: A Review». In: *Computers and Electronics in Agriculture* 151 (2018), pp. 61–69. DOI: 10.1016/j.compag.2018.05.012 (cit. on p. 7).

[43] A. Blum. «Crop responses to drought and the interpretation of adaptation». In: *Plant Growth Regulation* (1996) (cit. on p. 7).

[44] H. G. Jones. «Irrigation scheduling: advantages and pitfalls of plant-based methods». In: *Journal of Experimental Botany* (2004) (cit. on pp. 7, 8).

[45] H. G. Jones. «Monitoring plant and soil water status: established and novel methods revisited and their relevance to studies of drought tolerance». In: *Journal of Experimental Botany* (2007) (cit. on p. 8).

[46] N. C. Turner. «Techniques and experimental approaches for the measurement of plant water status». In: *Plant and Soil* (1981) (cit. on p. 8).

[47] K. A. Shackel. «The relationship between plant water status and physiological responses to water deficit». In: *Horticultural Reviews* (2011) (cit. on p. 8).

[48] S. A. O'Shaughnessy and S. R. Evett. «Canopy temperature based system effectively schedules and controls center pivot irrigation of cotton». In: *Agricultural Water Management* (2015) (cit. on p. 8).

[49] J. M. Blonquist, S. B. Jones, and D. A. Robinson. «A time domain transmission sensor with TDR performance characteristics». In: *Journal of Hydrology* (2005) (cit. on pp. 8, 9, 11).

[50] R. G. Allen, L. S. Pereira, D. Raes, and M. Smith. *Crop Evapotranspiration—Guidelines for Computing Crop Water Requirements*. FAO Irrigation and Drainage Paper 56. Food and Agriculture Organization of the United Nations, 1998, p. 300 (cit. on p. 8).

[51] H. G. Jones. «Use of thermography for quantitative studies of spatial and temporal variation of stomatal conductance over leaf surfaces». In: *Plant, Cell & Environment* 32.6 (2009), pp. 666–676 (cit. on p. 9).

[52] J. Penuelas et al. «Assessing plant water status by the reflectance and fluorescence remote sensing». In: *Journal of Experimental Botany* 44.259 (1993), pp. 1829–1835 (cit. on p. 9).

[53] Samuel O. Ihuoma and Chandra A. Madramootoo. «Recent advances in crop water stress detection». In: *Computers and Electronics in Agriculture* 141 (2017), pp. 267–275. ISSN: 0168-1699. DOI: 10.1016/j.compag.2017.07.026. URL: https://doi.org/10.1016/j.compag.2017.07.026 (cit. on p. 10).

[54] Laury Chaerle and Dominique Van Der Straeten. «Imaging techniques and the early detection of plant stress». In: *Trends in Plant Science* 5.11 (2000), pp. 495–501. DOI: 10.1016/S1360-1385(00)01781-7 (cit. on p. 10).

[55] Jan Behmann, Jörg Steinrücken, and Lutz Plümer. «Detection of early plant stress responses in hyperspectral images». In: *ISPRS Journal of Photogrammetry and Remote Sensing* 93 (2014), pp. 98–111. DOI: 10.1016/j.isprsjprs.2014.03.016. URL: http://dx.doi.org/10.1016/j.isprsjprs.2014.03.016 (cit. on p. 11).

[56] Ning Wang, Na Zhang, and Maohua Wang. «Wireless sensors in agriculture: A survey». In: *IEEE Sensors Journal* 8.1 (2018), pp. 308–319. DOI: 10.1109/JSEN.2008.915195 (cit. on p. 11).

[57] Marwan Ali Albahar. «A Survey on Deep Learning and Its Impact on Agriculture: Challenges and Opportunities». In: *Agriculture* 13.3 (2023). DOI: 10.3390/agriculture13030574. URL: https://doi.org/10.3390/agriculture13030574 (cit. on p. 12).

[58] Mor Soffer, Naftali Lazarovitch, and Ofer Hadar. «Real-Time Detection of Water Stress in Corn Using Image Processing and Deep Learning». In: *Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP)*. 2020. URL: https://doi.org/10.1109/ICIP2020.123456 (cit. on p. 12).

[59] Kavya Singh, Deepanshu Singh, and Nitin Mishra. «Review: Convolutional Neural Networks and Its Architecture». In: *International Journal of Health Sciences* 6.S1 (May 2022). DOI: 10.53730/ijhs.v6nS1.7074 (cit. on p. 13).

[60] Abhishek Narvaria, Uttam Kumar, Kanumuru Shree Jhanwwee, Anindita Dasgupta, and Gurdeep Kaur. «Classification and Identification of Crops Using Deep Learning with UAV Data». In: *2021 IEEE International India Geoscience and Remote Sensing Symposium (InGARSS)* (2021), pp. 153–156. URL: https://api.semanticscholar.org/CorpusID:249666647 (cit. on pp. 13, 14).

[61] Aleem Khaliq, Vittorio Mazzia, and Marcello Chiaberge. «Refining satellite imagery by using UAV imagery for vineyard environment: A CNN Based approach». In: *2019 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)* (2019), pp. 25–29. URL: https://api.semanticscholar.org/CorpusID:208206160 (cit. on pp. 13–15).

[62] Adduru U. G. Sankararao, Gattu Priyanka, Pachamuthu Rajalakshmi, and Sunitha Choudhary. «CNN Based Water Stress Detection in Chickpea Using UAV Based Hyperspectral Imaging». In: *2021 IEEE International India Geoscience and Remote Sensing Symposium (InGARSS)* (2021), pp. 145–148. URL: https://api.semanticscholar.org/CorpusID:249668122 (cit. on p. 13).

[63] Thomas A. Lake, Ryan D. Briscoe Runquist, and David A. Moeller. «Deep learning detects invasive plant species across complex landscapes using Worldview-2 and Planetscope satellite imagery». In: *Remote Sensing in Ecology and Conservation* 8 (2022). URL: https://api.semanticscholar.org/CorpusID:249685130 (cit. on p. 14).

[64] Mohd Hider Kamarudin, Zool Hilmi Ismail, Noor Baity Saidi, and Kousuke Hanada. «An augmented attention-based lightweight CNN model for plant water stress detection». In: *Applied Intelligence* (2023), pp. 1–16. URL: https://api.semanticscholar.org/CorpusID:258317091 (cit. on p. 14).

[65] Krit Rojanarungruengporn and Suree Pumrin. «Early Stress Detection in Plant Phenotyping using CNN and LSTM Architecture». In: *2021 9th International Electrical Engineering Congress (iEECON)* (2021), pp. 389–392. URL: https://api.semanticscholar.org/CorpusID:235308234 (cit. on p. 14).

[66] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. «Exploring the Limits of Weakly Supervised Pretraining». In: *arXiv preprint arXiv:1805.00932* (2018). URL: https://arxiv.org/abs/1805.00932 (cit. on p. 14).

[67] Jeremy Howard and Sebastian Ruder. «Universal Language Model Fine-tuning for Text Classification». In: *arXiv preprint arXiv:1801.06146* (2018). URL: https://arxiv.org/abs/1801.06146 (cit. on p. 15).

[68] Mehwish Moiz, Muh. Imadudin Akmal, Muhammad Shakeel Ishtiaq, and Usman Javed. «Classification of Rice Leaves Diseases by Deep CNN-Transfer Learning Approach for Improved Rice Agriculture». In: *2022 International Conference on Emerging Trends in Electrical, Control, and Telecommunication Engineering (ETECTE)*. 2022, pp. 1–6. URL: https://api.semanticscholar.org/CorpusID:255597597 (cit. on p. 15).

[69] Alaa Saeed, A. A. Abdel-Aziz, Amr Mossad, Mahmoud A. Abdelhamid, Alfadhl Y. Alkhaled, and Muhammad Mayhoub. «Smart Detection of Tomato Leaf Diseases Using Transfer Learning-Based Convolutional Neural Networks». In: *Agriculture* (2023). URL: https://api.semanticscholar.org/CorpusID:255652638 (cit. on p. 15).

[70] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. «A Survey on Deep Transfer Learning». In: *International Conference on Artificial Neural Networks*. 2018. URL: https://api.semanticscholar.org/CorpusID:51929263 (cit. on p. 15).

[71] Monu Bhagat, Dilip Kumar, and Sunil Kumar. «Optimized Transfer Learning Approach for Leaf Disease Classification in Smart Agriculture». In: *Multimedia Tools and Applications* 83 (2023), pp. 58103–58123. URL: https://api.semanticscholar.org/CorpusID:266406139 (cit. on p. 15).

[72] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. «A Comprehensive Survey on Transfer Learning». In: *Proceedings of the IEEE* 109 (2019), pp. 43–76. DOI: 10.1109/JPROC.2020.2992479. URL: https://doi.org/10.1109/JPROC.2020.2992479 (cit. on pp. 16, 24).

[73] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. «A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS». In: *Machine Learning and Knowledge Extraction* 5.4 (2023), pp. 1680–1716. ISSN: 2504-4990. DOI: 10.3390/make5040083. URL: https://www.mdpi.com/2504-4990/5/4/83 (cit. on p. 17).

[74] Dhanni Pavani, A. Nymisha, Nandini Reddy, and Neha Saw. «YOLO Object Detection Method for Underwater Trash Collecting Robot: Octacleaner». In: *2023 IEEE Technology & Engineering Management Conference - Asia Pacific (TEMSCON-ASPAC)*. 2023, pp. 1–5. URL: https://api.semanticscholar.org/CorpusID:269952877 (cit. on p. 20).

[75] Yanyu Guo, Xiao Hong Tian, and Yanting Xiao. «DBCR-YOLO: Improved YOLOv5 Based on Double-Sampling and Broad-Feature Coordinate-Attention Residual Module for Water Surface Object Detection». In: *Journal of Electronic Imaging* 32 (2023), pp. 043013–043013. URL: https://api.semanticscholar.org/CorpusID:259883644 (cit. on p. 20).

[76] Mino Sportelli, Orly Enrique Apolo-Apolo, Marco Fontanelli, Christian Frasconi, Michele Raffaelli, Andrea Peruzzi, and Manuel Pérez-Ruiz. «Evaluation of YOLO Object Detectors for Weed Detection in Different Turfgrass Scenarios». In: *Applied Sciences* (2023). URL: https://api.semanticscholar.org/CorpusID:260142982 (cit. on p. 20).

[77] Xiang Yue, Kai Qi, Xinyi Na, Yang Zhang, Yanhua Liu, and Cuihong Liu. «Improved YOLOv8-Seg Network for Instance Segmentation of Healthy and Diseased Tomato Plants in the Growth Stage». In: *Agriculture* (2023). URL: https://api.semanticscholar.org/CorpusID:261111698 (cit. on p. 20).

[78] Jiayao Zhuang, Xiaojun Jin, Yong Chen, Wenting Meng, Yundi Wang, Jialin Yu, and Bagavathiannan Muthukumar. «Drought Stress Impact on the Performance of Deep Convolutional Neural Networks for Weed Detection in Bahiagrass». In: *Grass and Forage Science* (2022). First published: 25 October 2022. DOI: 10.1111/gfs.12583. URL: https://doi.org/10.1111/gfs.12583 (cit. on p. 23).

[79] Sujata Butte, Aleksandar Vakanski, Kasia Duellman, Haotian Wang, and Amin Mirkouei. «Potato crop stress identification in aerial images using deep learning-based object detection». In: *Agronomy Journal* 113.4 (2021). Citations: 16, pp. 3476–3489. DOI: 10.1002/agj2.20841. URL: https://doi.org/10.1002/agj2.20841 (cit. on pp. 23, 31).

[80] Xiaohu Zhao, Jingcheng Zhang, Ailun Tang, Yifan Yu, Lijie Yan, Dongmei Chen, and Lin Yuan. «The Stress Detection and Segmentation Strategy in Tea Plant at Canopy Level». In: *Frontiers in Plant Science* 13 (2022), p. 949054. DOI: 10.3389/fpls.2022.949054. URL: https://www.frontiersin.org/articles/10.3389/fpls.2022.949054/full (cit. on p. 23).

[81] Connor Shorten and Taghi M. Khoshgoftaar. «A Survey on Image Data Augmentation for Deep Learning». In: *Journal of Big Data* 6 (2019), pp. 1–48. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0 (cit. on p. 24).

[82] Artur M. Gafurov, Svetlana S. Mukharamova, Anatoly Saveliev, and O. P. Yermolaev. «Advancing Agricultural Crop Recognition: The Application of LSTM Networks and Spatial Generalization in Satellite Data Analysis». In: *Agriculture* (2023). URL: https://api.semanticscholar.org/CorpusID:261131246 (cit. on p. 24).

[83] Arnauld Nzegha Fountsop, Jean Louis Ebongue Kedieng Fendji, and Marcellin Atemkeng. «Deep Learning Models Compression for Agricultural Plants». In: *Applied Sciences* (2020). URL: https://api.semanticscholar.org/CorpusID:225009467 (cit. on p. 24).

[84] Laith Alzubaidi et al. «Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions». In: *Journal of Big Data* 8 (2021). DOI: 10.1186/s40537-021-00444-8. URL: https://doi.org/10.1186/s40537-021-00444-8 (cit. on p. 24).

[85] G. Bradski. «The OpenCV Library». In: *Dr. Dobb's Journal of Software Tools* (2000) (cit. on p. 35).

[86] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, et al. «Array programming with NumPy». In: *Nature* 585.7825 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2 (cit. on p. 35).

[87] J. D. Hunter. «Matplotlib: A 2D Graphics Environment». In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55 (cit. on p. 35).

[88] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. «scikit-image: image processing in Python». In: *PeerJ* 2 (2014), e453 (cit. on p. 36).

[89] Pauli Virtanen et al. «SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python». In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on p. 36).

[90] Plotly Technologies Inc. *Collaborative data science*. 2015. URL: https://plot.ly (cit. on p. 36).

[91] Python Software Foundation. *os — Miscellaneous operating system interfaces*. https://docs.python.org/3/library/os.html (cit. on p. 37).

[92] Python Software Foundation. *shutil — High-level file operations*. https://docs.python.org/3/library/shutil.html (cit. on p. 38).

[93] Python Software Foundation. *random — Generate pseudo-random numbers*. https://docs.python.org/3/library/random.html (cit. on p. 38).

[94] Wes McKinney. «Data Structures for Statistical Computing in Python». In: *Proceedings of the 9th Python in Science Conference*. 2010, pp. 51–56. URL: https://pandas.pydata.org/papers/wesm-pandas-scipy-2010.pdf (cit. on p. 38).

[95] Alex Clark. *Pillow (PIL Fork) Documentation*. 2015. URL: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf (cit. on p. 38).

[96] Christoph Gohlke. *tifffile: Read and write TIFF files*. https://www.lfd.uci.edu/~gohlke/. 2020 (cit. on p. 38).

[97] Python Software Foundation. *pathlib — Object-oriented filesystem paths*. n.d. URL: https://docs.python.org/3/library/pathlib.html (cit. on p. 38).

[98] Python Software Foundation. *collections — Container datatypes*. n.d. URL: https://docs.python.org/3/library/collections.html (cit. on p. 38).

[99] Kirill Simonov. *PyYAML Documentation*. 2006. URL: https://pyyaml.org/wiki/PyYAML (cit. on p. 38).

[100] Fabian Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 38).

[101] Python Software Foundation. *glob — Unix style pathname pattern expansion*. n.d. URL: https://docs.python.org/3/library/glob.html (cit. on p. 38).

[102] Glenn Jocher, Rizwan Munawar, Ivor Zhu, and Laughing Q. *Ultralytics YOLO Format Documentation*. https://docs.ultralytics.com/datasets/detect/#ultralytics-yolo-format. Created 2023-11-12, Updated 2024-07-04. 2024 (cit. on p. 38).

[103] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009 (cit. on p. 39).

[104] Lutz Prechelt. «Early Stopping - But When?» In: *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 55–69 (cit. on p. 39).

[105] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006 (cit. on p. 39).

[106] Ron Kohavi. «A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection». In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. 1995 (cit. on p. 40).