CIWA

A WEB-BASED TOOL FOR AUTOMATED IRRIGATION MANAGEMENT USING INTELLIGENT PLANT CANOPY TEMPERATURE IDENTIFICATION TECHNIQUES

By

MINAS PANTELIDAKIS

A thesis submitted in partial fulfillment of
the requirements for the degree of

INTEGRATED MASTERS (5 YEAR DIPLOMA, 300 ECTS)

TECHNICAL UNIVERSITY OF CRETE
School of Electrical and Computer Engineering

TBD 2020

To the Faculty of Technical University of Crete:

The members of the Committee appointed to examine the thesis of MINAS PANTELIDAKIS find it satisfactory and recommend that it be accepted.

Georgios Chalkiadakis, Associate Professor

Athanasios A. Panagopoulos, Assistant Professor

Michail Zervakis, Professor

ACKNOWLEDGMENT

CIWA

A WEB-BASED TOOL FOR AUTOMATED IRRIGATION MANAGEMENT USING INTELLIGENT PLANT CANOPY TEMPERATURE IDENTIFICATION TECHNIQUES

Abstract

by Minas Pantelidakis, Electrical and Computer Engineering, Integrated Masters
Technical University of Crete
TBD 2020

Supervisor: Georgios Chalkiadakis

Canopy temperature has been recognized as a crop water stress indicator, since it reflects the interaction of crops with soil and the atmosphere. Even though there are several methods to estimate canopy temperature, most of them are time consuming, expensive, inaccurate, or require considerable human input. This work mobilizes Convolutional Neural Networks (CNNs) to identify sunlit leaves and in conjuction with thermal imagery, find the underlying leaf temperatures and calculate the Crop Water Stress Index (CWSI). The results of two different CNN architectures (FRRN, DeepLabV3) have been compared with two minimum input, state of the art methods, namely temperature Histogram Gradient Thresholding and Gaussian Mixture Models. We evaluate our approaches against this baseline using our own dataset of 1432 image-label pairs of pistachio trees. Results indicate that CNNs outperform existing methods. Our dataset is released for the scientific community to use. Finally, a web application was developed, so that researchers/growers can calculate the CWSI in real time, using pictures captured with a thermal camera.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dedication

This work is dedicated to my parents, Eugenia Theodoridou and Nikolaos Pantelidakis and to my sister, Fay Pantelidaki, who selflessly provided me with both financial resources and emotional support during my studies. I would like to thank the faculty and personnel of the Technical University of Crete and especially my supervisor Dr. Georgios Chalkiadakis. Special thanks to my co-supervisor Dr. Athanasions A. Panagopoulos for his tireless and insightful guidance throughout the implementation. Finally I would like to thank the Hellenic Airforce and specifically Squadron Leader Evangelos Papakostas for providing me with resources vital to the completion of this work.

Chapter One

Introduction

The San Joaquin Valley is a major agricultural production region in California. Nevertheless, water supplies can become increasingly limited in this region, due to prolonged droughts. The Valley is already in a water crisis and the primary challenge is to reduce consumption to restore the water balance between demand and renewable water supply. Farmers in the San Joaquin Valley are well aware of the water shortage problems and are actively looking for solutions to conserve water while maintaining production.

However, this phenomenon is not only observed in California. Freshwater scarcity is a universal issue threatening the growing world population and consequently the global food demand. Irrigated agriculture is the largest freshwater consumer, accounting for about 70 percent of freshwater withdrawals worldwide. Agricultural crop yields are higher in irrigated than in rain-fed farms, with the former being expected to increase in the future. Therefore, water scarcity is projected to impose an even greater threat in the years to come. Similarly, in Greece, the area actually irrigated increased from 932,980 hectares in 1990 to 1,142,180 hectares in 1995 and 1,294,400 hectares in 2003 [3].

Improved water management in irrigated farms is required to combat water scarcity. In general, irrigation water management can be improved through water application methods and irrigation scheduling. Water application methods include surface or gravity systems such as furrow and border irrigation, and pressurized irrigation systems, which include sprinkler

and drip (trickle) irrigation systems. Surface irrigation is by far the primary irrigation method worldwide. However, farmers have been changing their irrigation systems from the low-efficiency surface methods to the high-efficiency pressurized systems during the last few decades. Changing from the traditional surface to advanced pressurized irrigation systems is relatively costly, but it is a straightforward conversion for farmers to conserve water.

On the other hand, changing from traditional irrigation scheduling to advanced irrigation scheduling methods is not an easy task for most farmers worldwide. Irrigation scheduling is the most vital component of irrigation water management, and involves tasks like 'deciding when to irrigate' and 'how much water to apply' [27, 29]. Traditional irrigation scheduling methods are based on the condition of the crop, feel of soil, personal calendar, and when the neighbors begin to irrigate. Advanced or scientific irrigation scheduling methods include soil-based, plant-based, weather-based and model-based approaches.

A weather-based method typically uses weather data (e.g., solar radiation, air temperature, relative humidity, and wind speed) to estimate the evaporative demand of the atmosphere (reference evapotranspiration), combined with the related plant characteristics (crop coefficients) to estimate the water requirements of non-stressed plants. A soil-based method measures soil water content or potential and provides information on how much irrigation water would be required to recharge the soil profile. Weather and soil-based methods are being used for irrigation scheduling while making some assumptions for variables that are difficult to measure precisely in field conditions, such as stages of growth and related crop coefficients which vary between different crop species, growing conditions, depths of root area, physical texture in different soil layers and soil readily available water levels.

Plant-based methods, on the other hand, measure plant water status directly and provide information on the timing of irrigation. Plant-based methods have the potential to be more sensitive and accurate than weather- and soil-based methods and are more suitable for precision and regulated deficit irrigation practices. Existing data indicate that farmers, even in advanced countries such as the United States of America (U.S.), are still using traditional

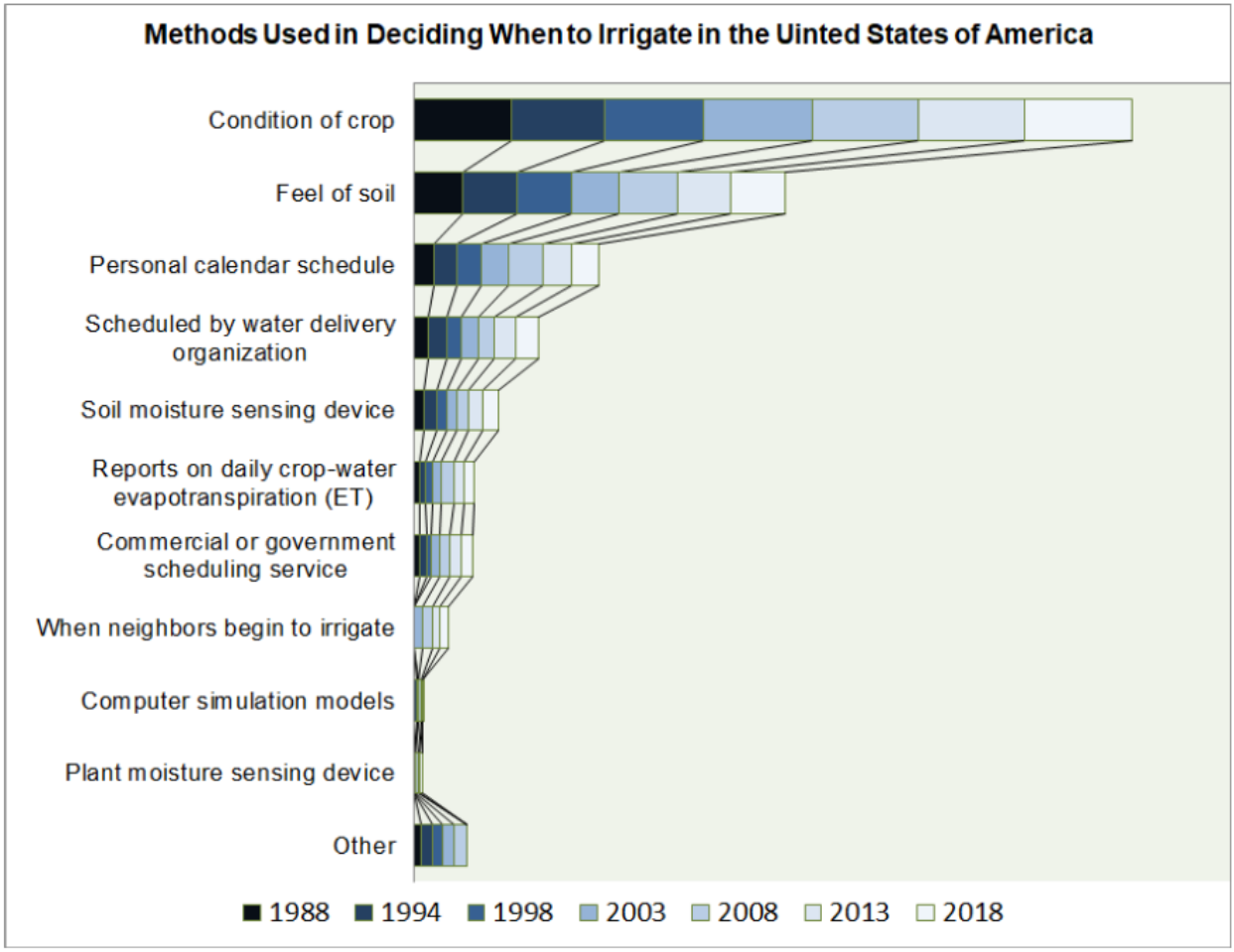


Figure 1.1 Methods used in deciding when to irrigate in the U.S.A.

methods to schedule irrigation [24]. As shown in Figure 1.1, farmers in the U.S. are mostly using the condition of the crop, feel of soil, and personal calendar to decide 'when to irrigate'.

Switching from low-efficiency traditional methods to high-efficiency scientific irrigation scheduling methods would result in substantial benefits worldwide, such as conserving fresh-water supplies and improving crop yield and quality. Additionally, the scientific irrigation scheduling methods contribute to reduced water application, thus resulting in less leaching of synthetic fertilizer and contamination of groundwater supplies.

Precision and deficit irrigation scheduling create great opportunities for water saving applications. The main objective of this work is to use the problem-solving power of Artificial Intelligence (AI) and machine learning principles to develop an automated and easy-to-use

plant-based canopy temperature method with minimal human intervention for monitoring water stress in seed trees.

In nature, all objects emit infrared energy as a function of their surface temperature. Canopy temperature has long been recognized as an indicator of plant water status [36, 52], and numerous studies have shown the suitability of canopy temperature to detect plant water stress [23, 25, 22, 32, 33, 31]. When water is limited, plant transpiration decreases, and plant temperature increases due to the decreased cooling mechanism. Plant temperature technology had not advanced enough in the past due to (1) lack of suitable and affordable infrared cameras to measure the plant temperature, and (2) difficulty to capture representative images for plant water status, as infrared cameras will give the average of everything within their field of view (sunlit leaves, shaded leaves, tree trunk, voids in canopy). Today, with low-cost wireless infrared thermometers available, plant temperature technology is within reach for commercial applications. One of the tasks of this work is to address the second drawback of plant temperature technology: *to develop a method that automatically identifies candidate (healthy sunlit) leaves for plant temperature estimation.*

Many approaches have been proposed to automate the identification of canopy temperature in plants e.g. [54, 53, 34]. In particular, the work of [54] developed a method for aligning infrared (IR) thermography images with regular images in grapevines. The method is motivated by and suitable for extracting the canopy temperature from the aligned images. Nevertheless, an algorithm is not proposed to this end. On the same line of research, the work of [53] proposes a system that builds on top of [54] and utilizes the aligned images to identify the canopy temperature of leaves. The work relies on simple color identification techniques and Gaussian mixture distribution extraction techniques to identify the leaves and extract the canopy temperature, respectively. The method is "successful when the underlying leaf area in the IR image can be reasonably identified" [54].

Recently, solutions based on advanced image recognition techniques have also been emerging. Image recognition preoccupies itself with recognizing patterns in digital images with

minimum to zero human intervention. It is an interdisciplinary subfield of machine vision and artificial intelligence, and brings together statistical, machine learning, pattern recognition, and image processing techniques. In recent years, advancements in image recognition and, most importantly in deep convolutional neural networks and the corresponding implementations for image recognition tasks, have led to inspiring results in the field [49]. In this context, such approaches are actively being investigated for tasks involving leaf recognition for agricultural and related activities [35]. For instance, the work of [55] proposes a convolutional neural network architecture and data pre-processing techniques to classify leaves according to the tree they originate from.

Similarly, the work of [57] proposes a technique for plant leaf identification via a growing convolutional neural networks with progressive sample learning [6]. However, all these works do not focus on canopy temperature identification. In addition, most of these works consider single leaves in isolation and, hence, do not deal with the challenges of detecting multiple leaves in an image and identifying the sunlit ones. Estimating the canopy temperature from images requires to first detect the sunlit leaves and separate them from the leaves below as well as the ground, branches and other potential objects, such as animals and human artifacts. Such a complex task of detecting multiple objects in noisy images is an active area of research in image recognition and a profound challenge [19]. In this respect, the work most relevant to ours is the work of [48], which preoccupies itself with identifying multiple overlapping leaves. Although the approach does not generalize to purely illuminated or noisy images (for instance, ones that also include branches and land), it illustrates the potential of convolutional neural networks and standard edge detection.

Against this background, we propose a Convolutional Neural Network (CNN) method to identify candidate (healthy sunlit) leaves for plant canopy temperature estimation and calculate the Crop Water Stress Index (CWSI). The proposed method is non-intrusive, requires minimum user input, performs in almost real time and achieves better results than previously used methods. The chosen CNN architectures are Full-Resolution Residual Net-

work (FRRN) and DeepLabV3. Both these CNN implementations perform well on the task of semantic image segmentation. The case study of the mentioned method is pistachio. We created our own dataset of pistachio tree images to test the proposed and baseline methods against. Finally we developed a web application for precision irrigation.

In a nutshell, our main contributions are:

- The utilization and testing of two semantic-segmentation-capable CNN architectures for sunlit leaf identification in noisy images.
- The collection, labeling, and release of the first semantic image segmentation dataset of pistachio sunlit leaves (1432 image-label pairs of metadata-enriched visible and infrared spectrum pistachio images)
- The development of a modern, easy-to-use, responsive web application for precision irrigation.

The rest of this thesis is structured as follows: In Chapter 2 we discuss the crop water stress index, gaussian mixture models, convolutional neural networks, semantic image segmentation and provide some background on web development basics. In Chapter 3 we discuss our approach; we examine FRRN and DeepLabV3 in more detail, present the selected baseline methods, and elaborate on the data acquisition process, equipment and labeling process. In Chapter 4 we compare against the tested methods and showcase the effectiveness of our approach. In Chapter 5 we present the implemented web application and illustrate its functionality. In Chapter 6 we conclude this work, discussing the limitations of our approach and future work.

Chapter Two

Related Work

2.1 Agricultural background

2.1.1 FAO Irrigation Scheduling Method

One of the scientific irrigation scheduling methods is presented by the Food and Agricultural Organization of the United Nations (FAO) in 1998 [2]. This FAO guideline, which is a global standard and has been utilized by countless researchers worldwide, is based on crop evapotranspiration (ETc) and soil water balance as follows:

$$D_{r,i} = D_{r,i-1} - (P - RO)_i - I_i - CR_i + ETc_{act,i} + DP_i \tag{2.1}$$

where:

$D_{r,i}$	Root zone depletion at the end of day i[mm]
$D_{r,i-1}$	Water content in the root zone at the end of the previous day $i-1$[mm]
P_i	Precipitation on day i[mm]
RO_i	Runoff from the soil surface on day i[mm]
I_i	Net irrigation depth on day i that infiltrates the soil[mm]
CR_i	Capillary rise from the groundwater table on day i[mm]
$ETc_{act,i}$	Actual crop evapotranspiration on day i[mm]
DP_i	Water loss out of the root zone by deep percolation on day i[mm]

Crop evapotranspiration is the sum of transpiration and evaporation from plant and soil surfaces. When soil water is adequate, ET_c occurs at potential rates. In nature and irrigated farms, however, soil water is not always adequately available. In that case, actual ET_c rates are less than potential rates. Actual Et_c rates occur under real field conditions and are very difficult to estimate accurately. The actual ET_c is the most critical unknown in the soil water balance study or any hydrological water balance studies for that matter. When the soil water content is not adequate, the adjusted ET_c can be estimated as follows[2, 28]:

$$ETc_{actual} = (Ks \times Kcb + Ke) \times ETo \tag{2.2}$$

where ETo is reference evapotranspiration and represents the atmospheric water demand; Kcb is the basal crop coefficient, representing the crop type and stage of growth; Ke is the soil evaporation coefficient, representing the surface wetness of the soil; and Ks is the crop water stress coefficient, which is a dimensionless transpiration reduction factor. Where soil water is in limiting conditions, Ks < 1, and where soil water is adequate, Ks=1.

Equation 2.2 is very popular in irrigation research communities and industries, and many countries, local governments and research institutes publish the daily ETo values through their weather monitoring networks. However, one of the challenges equation 2.2 imposes is the accurate estimation of the crop water stress coefficient. In FAO 56 [2], Ks is given as:

$$Ks = \frac{TAW - Dr}{TAW - RAW} = \frac{TAW - Dr}{(1-p) \times RAW} \tag{2.3}$$

where TAW is the total depth of available soil water in the root zone (mm), Dr is the root zone depletion (mm), RAW is the depth of readily available water in the root zone (mm), and p is the fraction of TAW that a crop can extract from the root zone without suffering water stress. When soil water depletion (Dr) is less than or equal to the readily available water (RAW), there is no crop water stress and Ks=1.0. The determination of the crop water stress coefficient (Ks) is crucial in precision irrigation scheduling to conserve water and at the

same time avoid any crop yield and quality losses. However, the soil-based Ks is complicated to determine in commercial farms due to the spatial variability of soil characteristics such as soil type, texture and salinity, and variations of TAW, Dr and p over the growing season. In general, plant-based stress coefficients or parameters provide a direct and better indication of crop water stress.

2.1.2 Crop Water Stress Index

Plant water stress is a condition that plants are subject to when soil water is limiting, and transpiration and photosynthesis fall below the potential rates, thus inhibiting plant growth and production. There are several plant water stress indices in irrigation research communities [9, 11], but the Crop Water Stress Index (CWSI) has received the most attention. Crops with adequate soil water transpire at the potential rate for the prevailing environmental conditions. As the soil water becomes limiting, the actual transpiration will be less than the potential rate. The CWSI is a measure of the ratio of actual to potential transpiration. Plant canopy temperature is a direct indication of plant water status and can be directly used to detect crop water stress for irrigation scheduling purposes. Transpiration is the primary cooling mechanism in plants. When soil water is limiting, the transpiration rate decreases, the available energy is converted to sensible heat and leaf temperature increases. Irrigation management using remotely sensed canopy temperature is becoming more common using satellite, aerial, and ground-based platforms.

Jackson et al.[26] and Idso et al.[23] introduced, respectively, the theoretical and empirical CWSI. The general CWSI equation can be shown as:

$$CWSI = \frac{(T_c - T_a)_M - D_1}{D_2 - D_1} \tag{2.4}$$

where Tc is the crop canopy temperature (° C), Ta is the air temperature (° C), and subscript M denotes "measured". D1 is the no-stressed lower level $(T_c - T_a)_{LL}$ and D2 is a complete

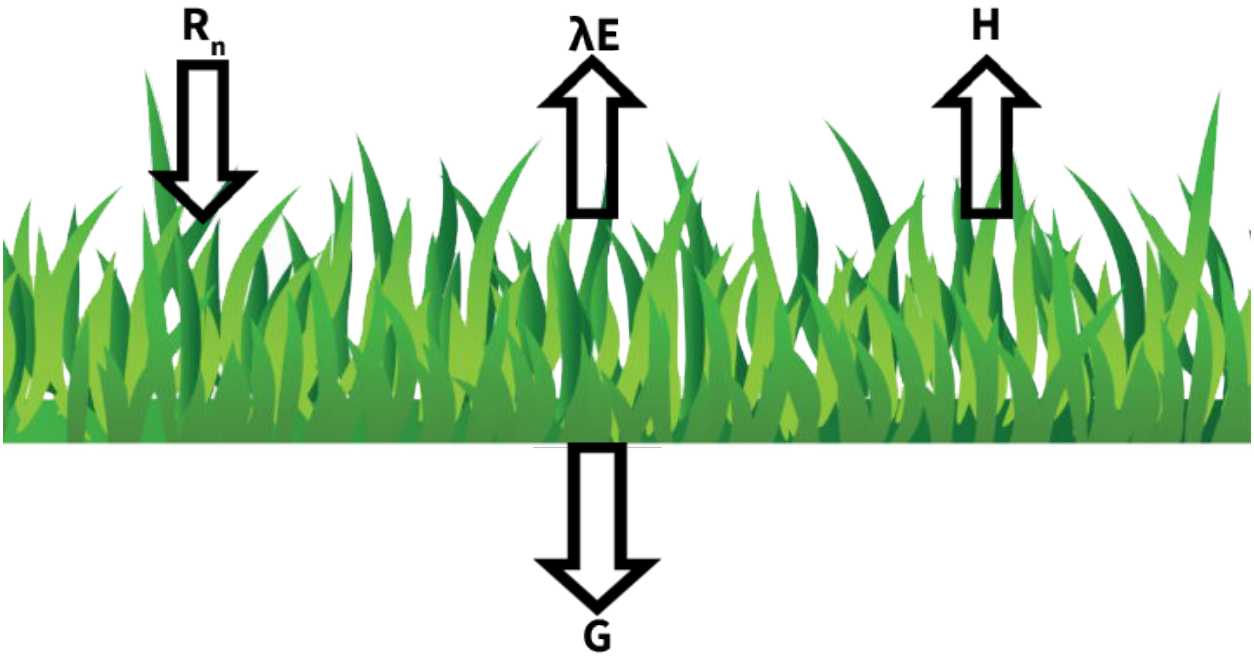


Figure 2.1 Schematic showing primary components of the energy balance.

stress upper level $(T_c - T_a)_{UL}$. The CWSI concept has been used for irrigation scheduling and water management by numerous researchers worldwide [40, 51, 4, 18, 39, 50, 17].

2.1.3 CWSI and Energy Balance Considerations

The CWSI is based on the crop energy balance and the determinations of baselines D1 and D2 in Equation 2.4 are crucial to computing CWSI. The energy balance for a crop canopy is illustrated in Figure 2.1 [28] and can be expressed as follows:

$$R_n = \lambda E + H + G \tag{2.5}$$

where R_n is net radiation $(MJm^{-2}d^{-1})$, λE is latent heat flux $(MJm^{-2}d^{-1})$, E is evotranspiration, λ is the latent heat of vaporization $(MJkg^{-1})$, H is sensible heat flux $(MJm^{-2}d^{-1})$ and G is soil heat flux $(MJm^{-2}d^{-1})$.

The soil heat flux is usually small and can be ignored. The latent heat flux and sensible

heat flux can be computed as:

$$\lambda E = \frac{\rho_a c_p (e_s - e_a)}{\gamma (r_{av} + r_s)} \tag{2.6}$$

$$H = \frac{\rho_a c_p (T_o - T_a)}{r_{ah}} \tag{2.7}$$

where ρ_a is the air density (kgm^{-3}), c_p is the specific heat capacity of air ($kJkg^{-1\circ}C^{-1}$), e_s is saturation vapor pressure (kPa), e_a is actual vapor pressure (kPa), γ is the psychrometric constant ($kPaC^{-1}$), r_{av} is aerodynamic resistance to turbulent vapor transfer (sm^{-1}), r_s is bulk canopy resistance to vapor flow (sm^{-1}) and r_{ah} is the aerodynamic resistance for sensible heat flux.

Equations 2.5,2.6,2.7 are the foundation of the well-known Penman-Monteith equation [28], which is used to estimate ETc. Penman-Monteith equation is recommended by the FAO [2] and used as standard for ETo estimation worldwide. As introduced by Jackson et al. [26], combining equations 2.5,2.6 and 2.7, defining Δ as the slope of saturation vapor pressure vs. temperature, $\Delta = \frac{e_s - e_a}{T_c - T_a}$ in units of $kPaC^{-1}$ and solving for $T_c - T_a$ yields:

$$T_c - T_a = [\frac{r_a (R_n - G)}{\rho_a c_p}][\frac{\gamma (1 + \frac{r_c}{r_a})}{\Delta + \gamma (1 + \frac{r_c}{r_a})}] - [\frac{e_s - e_a}{\Delta + \gamma (1 + \frac{r_c}{r_a})}] \tag{2.8}$$

The lower limit of $(T_c - T_a)$, or D_1 can be found from equation 2.8 when soil water is freely available and crop resistance approaches zero:

$$D_1 = (T_c - T_a)_{LL} = [\frac{r_a (R_n - G)}{\rho_a c_p}][\frac{\gamma}{\Delta + \gamma}] - [\frac{e_s - e_a}{\Delta + \gamma}] \tag{2.9}$$

The upper limit of $(T_c - T_a)$, or D_2 can be found from equation 2.8 when soil water is severely limited and crop resistance approaches infinity ($r_c \to \infty$):

$$D_2 = (T_c - T_a)_{UL} = [\frac{r_a (R_n - G)}{\rho_a c_p}] \tag{2.10}$$

The theoretical CWSI [26] requires measurements of canopy temperature (T_c), air temperature (T_{air}), relative humidity (RH), net radiation, soil heat flux and aerodynamic resistance. Such measurements are complicated to obtain in commercial farms. The theoritical CWSI is suitable for research projects, but not for farm applications, due to the fact that it requires extensive measurements. Idso et al. [23] conducted several experiments on different crops at different locations and collected crop foilage temperature, air temperature and humidity data. They showed that plots of $T_c - T_a$ vs. vapor pressure deficit (VPD) of well-watered plants, transpiring at potential rate, yield linear relationships under clear sky during daytime. They developed an empirical CWSI that is reasonably independent of environmental variability.

The empirical CWSI is suitable for farm applications, as it only requires canopy temperature (T_c), air temperature (T_{air}), relative humidity (RH). The empirical CWSI has been employed by numerous researchers for irrigation scheduling and water management [40, 51, 4, 18, 39, 50, 17].

The CWSI formulation (Equation 2.4) is shown in Figure 2.2 for graphical calculations. When soil water is limited and crop ET falls below the potential rate, crop canopy temperature increases and the $T_c - T_a$ vs. VPD data point will be located somewhere above the non-water stress baseline X in Figure 2.2, for example. Thus, as the ratio of actual transpiration to potential transpiration goes from 1 to 0, the CWSI goes from 0 to 1 [9, 23].

In the empirical CWSI, the relationship between $T_c - T_a$ and VPD — during the day, around solar noon and under clear sky — is used to establish a non-water-stressed baseline (D_1) for different crops. The vapor pressure deficit is the difference between vapor pressure at saturation (the maximum water vapor the air can hold) and actual vapor pressure and represents the atmospheric water demand for vegetation. Thus, a VPD of zero indicates no water demand and corresponds to 100% RH. The severely-stressed baseline (D_2) is estimated as the difference between saturated vapor pressure at air temperature (e_s) and saturation vapor pressure at air temperature plus the intercept value, as shown in the following equations:

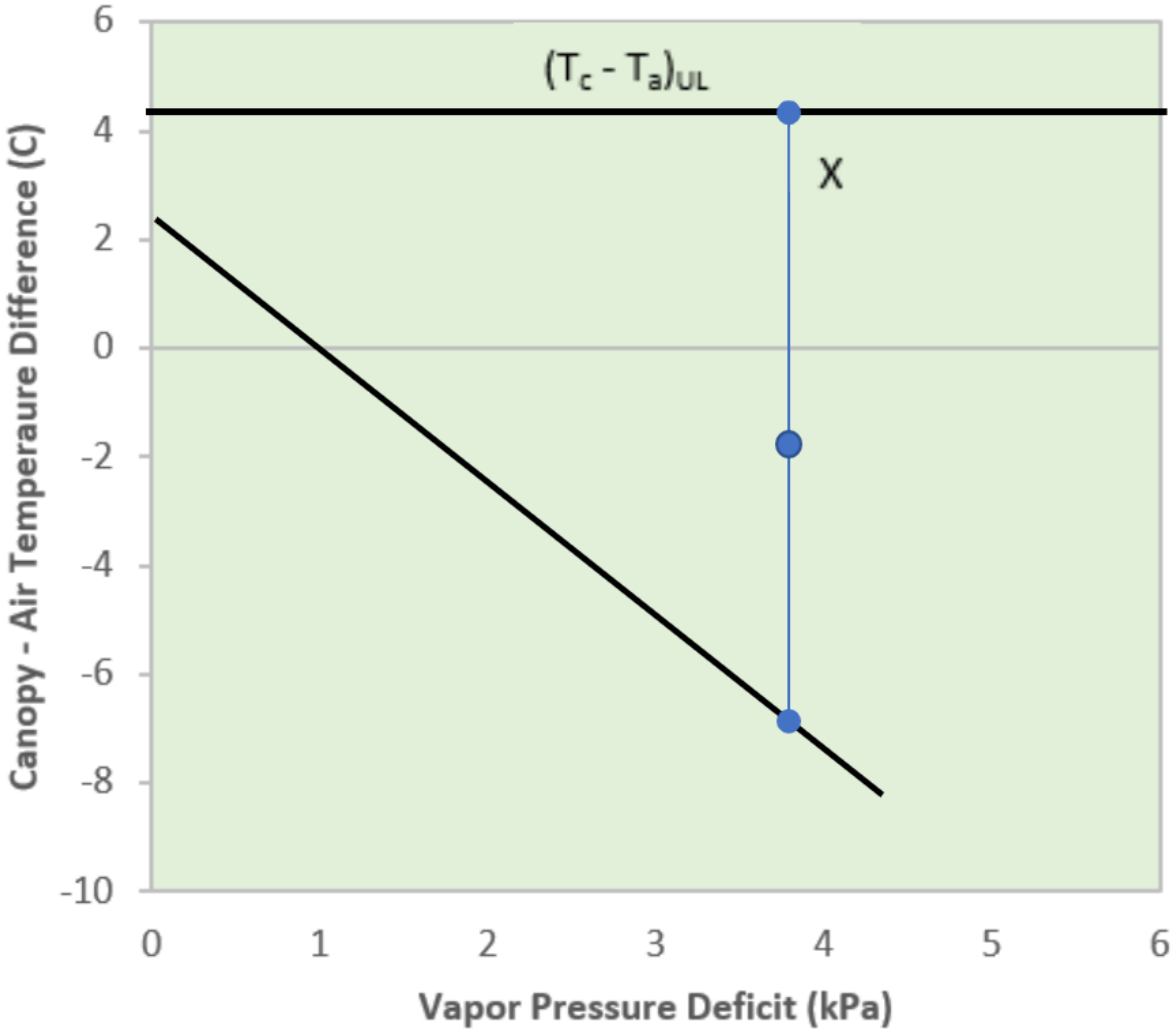


Figure 2.2 Graphical representation of lower and upper baselines, CWSI = YZ/XZ

$$D_1 = m \times VPD + b \tag{2.11}$$

$$D_2 = m \times VPG + b \tag{2.12}$$

$$VPD = e_s - e_a \tag{2.13}$$

$$VPG = e(T_A) - e(T_A + b) \tag{2.14}$$

where b, m, VPD and VPG are, respectively, intercept, slope, vapor pressure deficit (kPa), and vapor pressure gradient (kPa). Furthermore, e_s is saturation vapor pressure at air temperature (kPa), e_a is the actual vapor pressure of air (kPa), i.e, $e_a = e_s \times RH$.

2.1.4 CWSI and Ks Relationship

Potential or standard transpiration of crops can be estimated using available guidelines and data [2, 28]. Actual transpiration, however, is difficult to estimate under field conditions. The CWSI can be used to estimate the actual transpiration [26].

$$Actual\ Transpiration = (1 - CWSI) \times Standard\ Transpiration \tag{2.15}$$

As mentioned previously, the determination of Ks of the FAO's equation 2.2 is problematic in commercial farms. The CWSI concept can be used to determine real-time Ks under actual field conditions:

$$Ks = 1 - CWSI \tag{2.16}$$

When soil water is adequate and there is no water stress, CWSI approaches zero. For severely-stressed crops, CWSI approaches unity. The FAO's equation 2.2 can be expressed in terms of CWSI as:

$$ETcactual = [(1 - CWSI)Kcb + Ke]ETo \tag{2.17}$$

Equation 2.17 can be used with local ETo values to estimate actual crop evotranspiration, which is required in precision irrigation scheduling.

2.2 Computer Science background

2.2.1 Gaussian Mixture Models (GMMs)

Gaussian Mixture Models, conceived by Richard Duda and Peter Hart [12], is a soft clustering method, meaning that apart from a description of the clusters themselves, it also provides an uncertainty measure or probability that indicates how much a data point is associated with a particular cluster.

As Gaussian Mixture (Figure 2.3) we define a function that is comprised of multiple Gaussians, each identified by $c \in \{1, \ldots, C\}$, where C is the number of components (clusters) of the dataset. Each Gaussian c of a mixture is specified by the following parameters:

- A mean μ that defines its centre.
- A covariance Σ that defines its width.
- A mixing probability π that defines the size of the Gaussian function.

The following explanation of Gaussian Mixture models is heavily influenced by https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95, which provides an excellent description of the method.

The mixing coefficients of the Gaussians are probabilities and must meet the following condition:

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2.18}$$

To determine the optimal values for these parameters we calculate the maximum likelihood[38]. The multivariate Gaussian density function is given by:

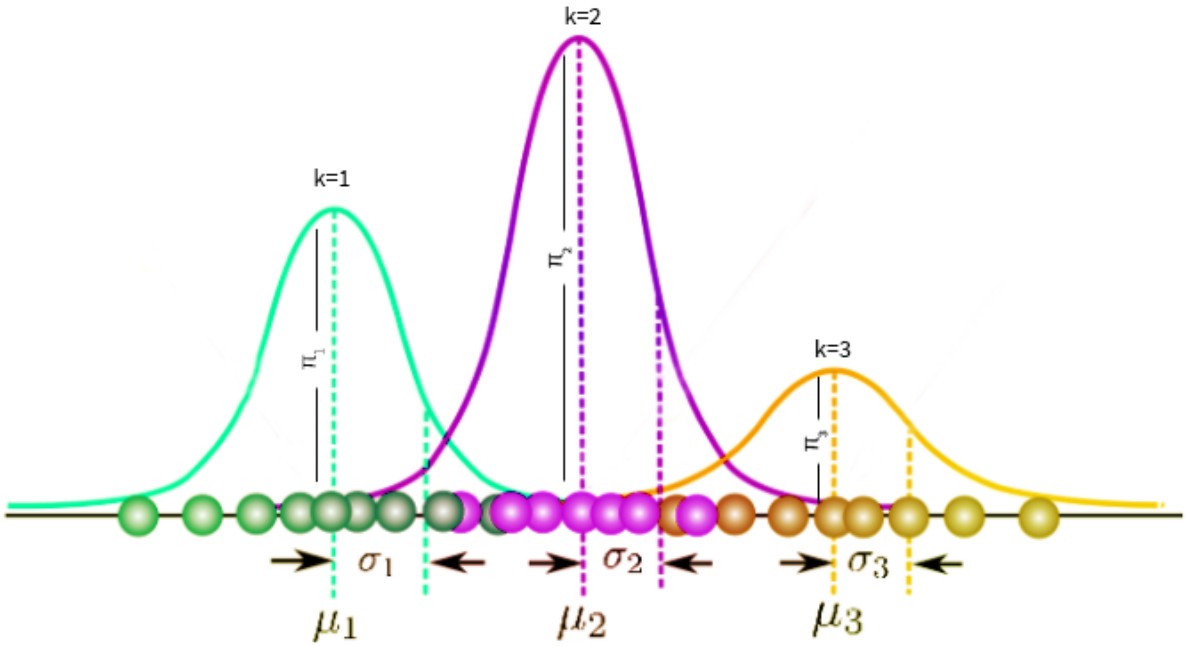


Figure 2.3 Gaussian Mixture with K = 3.

$$N(x|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{2.19}$$

For x data point with D dimensions. μ and Σ represent the mean and covariance of the distribution. It is also useful to take the log of this equation, which is given by:

$$\ln N(x|\mu,\Sigma) = -\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln\Sigma - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) \tag{2.20}$$

Differentiating equation 2.20 with respect to μ and Σ and then equating to zero, one is able to find the optimal values for these parameters. The solutions will correspond to the Maximum Likelihood Estimates (MLE). However, we need to find the parameters for the whole mixture, and not just one Gaussian.

The probability that a data point x_n comes from some Gaussian k is expressed as:

$$p(z_{nk} = 1|x_n) \tag{2.21}$$

where z is a latent(hidden) variable with two possible values. It is equal to one if x comes from Gaussian k, and zero otherwise. Likewise one can state the following:

$$\pi_k = p(z_k = 1) \tag{2.22}$$

Meaning that the probability of observation for a data point that belongs to Gaussian distribution k is equivalent to the mixing coefficient for k. Now let z be the set of all possible latent variables z, hence:

$$\boldsymbol{z} = \{z_1, \ldots, z_K\} \tag{2.23}$$

It is known that each z is independent of others and it can only be equal to one when k is equal to the cluster the point comes from. Therefore:

$$p(\boldsymbol{z}) = p(z_1 = 1)^{z_1} p(z_2 = 1)^{z_2} \ldots p(z_K = 1)^{z_K} = \prod_{k=1}^{K} \pi_k^{z_k} \tag{2.24}$$

The probability of observing our data given that it came from Gaussian k is the Gaussian function itself. Following the same logic we used to define p(z), we can state:

$$p(x_n|\boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}(x_n|\mu_k, \Sigma_k)^{z_k} \tag{2.25}$$

The equations we have just derived, along with the Bayes rule, will help us determine the probability of z given our observation x. From the product rule of probabilities, we know that

$$p(x_n, \boldsymbol{z}) = p(x_n|\boldsymbol{z})p(\boldsymbol{z}) \tag{2.26}$$

The operands on the right are what we have just found. However, before we use the Bayes rule we will need to find $p(x_n)$, not $p(x_n, z)$. To get rid of z we just need to sum up the terms on z (marginalization), hence:

$$p(x_n) = \sum_{k=1}^{K} p(x_n|\boldsymbol{z})p(\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \tag{2.27}$$

This is the equation that defines a Gaussian Mixture. To determine the optimal values for its parameters, we need to determine the maximum likelihood of the model. We can find the likelihood as the joint probability of all observations x_n, defined by:

$$p(\boldsymbol{X}) = \prod_{n=1}^{N} p(x_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \tag{2.28}$$

Applying the log to each side of the equation we get:

$$\ln p(\boldsymbol{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \tag{2.29}$$

From Bayes rule, we know that:

$$p(z_k = 1|x_n) = \frac{p(x_n|z_k = 1)p(z_k = 1)}{\sum_{j=1}^{K} p(x_n|z_j = 1)p(z_j = 1)} \tag{2.30}$$

From our earlier derivations we learned that:

$$p(z_k = 1) = \pi_k,$$
$$p(x_n|z_k = 1) = \mathcal{N}(x_n|\mu_k, \Sigma_k)$$

Replacing the above in equation 2.30 we get:

$$p(z_k = 1|x_n) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} = \gamma(z_{nk}) \tag{2.31}$$

That $\gamma(z_{nk})$ is sometimes referred to as a responsibility value, since it measures the probability that a data point x_n belongs to cluster k.

On the right side of equation 2.29 there is a logarithm that makes calculating the derivative of the expression hard. To estimate the parameters for the Gaussian Mixture we use an iterative method called Expectation Maximization (EM).

The Expectation - Maximization (EM) algorithm

Expectation - Maximization (EM) is widely used for optimization problems. It can be mobilized as an alternative to gradient descent with the advantage that, frequently, the updates can be computed analytically.
Let the parameters of our model be:

$$\theta = \{\pi, \mu, \Sigma\}$$

Given the above model, the EM algorithm would perform the following steps [5]:

Step 1: Initialise θ. The initialization could be random, or the result of another model.

Step 2 (Expectation step):

$$Q(\theta^*, \theta) = E[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\theta^*)] = \sum_{Z} p(\boldsymbol{Z}|\boldsymbol{X}, \theta) \ln p(\boldsymbol{X}, \boldsymbol{Z}|\theta^*) \tag{2.32}$$

The $p(\boldsymbol{Z}|\boldsymbol{X}, \theta)$ of equation 2.32 is $\gamma(z_{nk})$ of equation 2.31, so after substitution we get:

$$Q(\theta^*, \theta) = \sum_{Z} \gamma(z_{nk}) \ln p(\boldsymbol{X}, \boldsymbol{Z}|\theta^*) \tag{2.33}$$

We are still missing $p(\boldsymbol{X}, \boldsymbol{Z}|\theta^*)$, which is the complete likelihood of the model, including both X and Z. We can find it by using the following expression:

$$p(\boldsymbol{X}, \boldsymbol{Z}|\theta^*) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi^{z_{nk}} \mathcal{N}(x_n|\mu_k, \Sigma_k)^{z_{nk}} \tag{2.34}$$

Which is the result of calculating the joint probability of all observations and latent variables and is an extension of our initial derivations for p(x). The log of this expression is given by:

$$\ln p(\boldsymbol{X}, \boldsymbol{Z}|\theta^*) = \prod_{n=1}^{N} \prod_{k=1}^{K} z_{nk}[\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Sigma_k)] \tag{2.35}$$

We can replace 2.35 in 2.33 to get:

$$Q(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Sigma_k)] \tag{2.36}$$

In the maximization step, we will find the revised parameters of the mixture. For this purpose, we will need to make Q a restricted maximization problem and thus we will add a Lagrange multiplier to equation 2.36.

Step 3 (Maximization step): Find the revised parameters θ^* using:

$$\theta^* = argmax_\theta Q(\theta^*, \theta)$$

Where $Q(\theta^*, \theta)$ is described by equation 2.36. However, Q should also take into account the restriction that all π values should sum up to one. To do so, we will need to add a suitable Lagrange multiplier. Therefore, we should rewrite Equation 2.36 in this way:

$$Q(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk})[\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Sigma_k)] - \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \tag{2.37}$$

And now we can easily determine the parameters by using maximum likelihood. Let's now take the derivative of Q with respect to π and set it equal to zero:

$$\frac{\partial Q(\theta^*, \theta)}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\gamma(z_{nk})}{\pi_k} - \lambda = 0 \tag{2.38}$$

Then, by rearranging the terms and applying a summation over k to both sides of the equation, we obtain:

$$\sum_{n=1}^{N} \gamma(z_{nk}) = \pi_k \lambda \Rightarrow \sum_{k=1}^{K} \sum_{n=1}^{N} \gamma(z_{nk}) = \sum_{k=1}^{K} \pi_k \lambda \tag{2.39}$$

From 2.18, we know that the sum of all mixing coefficients π equals to one. In addition, we know that summing up the probabilities γ over k will also give us one. Thus we get $\lambda = N$. We then can solve for π:

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N} \tag{2.40}$$

Similarly, if we differentiate Q with respect to μ and Σ, equate the derivative to zero and solve for the parameters using the log-likelihood equation 2.20, we get:

$$\mu_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{2.41}$$

$$\Sigma_k^* = \frac{\Sigma_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{2.42}$$

These revised values will be used to calculate γ in the following EM iteration and so forth until the likelihood value converges.

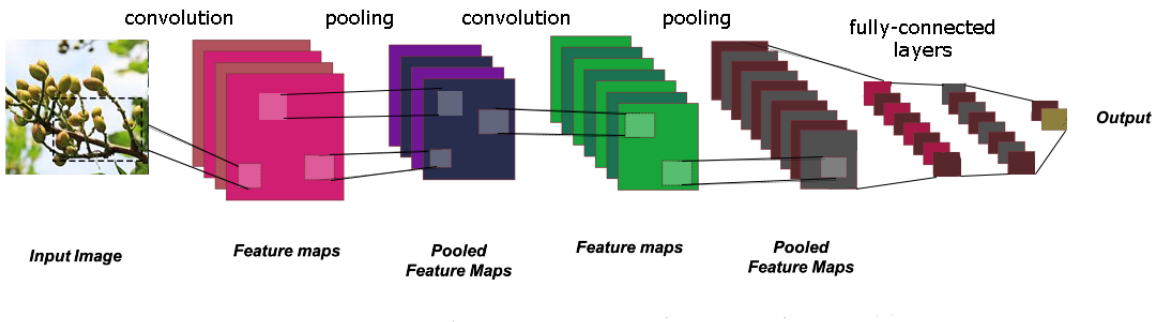


Figure 2.4 Typical CNN architecture.

2.2.2 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is an artificial neural network that has been popularly used for image analysis, but can also be used for other data analysis and classification tasks. One can think of a CNN as an artificial neural network that has some kind of specialization, enabling it to pick out or detect patterns and make sense of them. The full architecture of a typical CNN can be seen in Figure 2.4.

Convolutional layers and feature extraction

A CNN also has convolutional layers, which are hidden. Apart from convolutional layers, CNNs typically have non-convolutional layers as well, but their name is given after the convolutional layers. A convolutional layer accepts input, performs some kind of convolutional operation on it and forwards the output to the next layer. Convolutional layers are used for pattern detection and have a set number of filters (kernels) specified. Kernels are what actually identifies different patterns.

Kernels can be imagined as small matrices with predefined number of columns and rows. The filter convolves (slides) across the input, with the output being the dot product of the input-filter combination. Filters could detect multiple edges, shapes, textures, objects etc. For example, if a filter detects edges, it's called an edge detector. Some filters might detect

particular shapes like squares, corners, circles. etc. This kind of filters (geometric filters) are typically encountered at the early layers of the network. As the network goes deeper, these filters become more elaborate. In deeper layers, kernels might be able to detect specific shapes and simple objects. In even deeper layers kernels are capable of detecting even more complex entities like full animals, cars, humans etc. This process generates feature maps, since certain features of the input are extracted by the convolution filters.

Pooling layers

Feature maps are susceptible to localization issues, while ideally, features should be location agnostic. To prevent these kind of issues, several techniques are employed. One technique is feature map down-sampling. This technique achieves more adaptive feature maps, that are not confused by a possible feature transformation (position, scale, rotation) in an image.

This down-sampling is performed by feeding the feature maps into pooling layers. Pooling layers summarize the feature by highlighting its key elements. Two commonly employed pooling methods are max-pooling and average-pooling, that descriptively summarize the most dominant and the average presence of a feature respectively.

Adding pooling layers after convolutional layers is a common practice used in CNNs. There might be multiple pooling layers in a CNN. Pooling layers output the same number of feature maps that are fed into them.

Fully Connected Layers

The output of the convolution - pooling combination, is flattened into an 1-dimensional vector and fed into a fully-connected unit that determines the classification decision. Each value of this vector represents the probability that a given feature corresponds to a label.

Figure 2.5 depicts the way inputs are fed into the first layer, where they are multiplied by weights and pass through an activation function (usually Rectified Linear Unit (ReLu) [16]). They then pass forward to the output layer. Each neuron of the output layer represents a

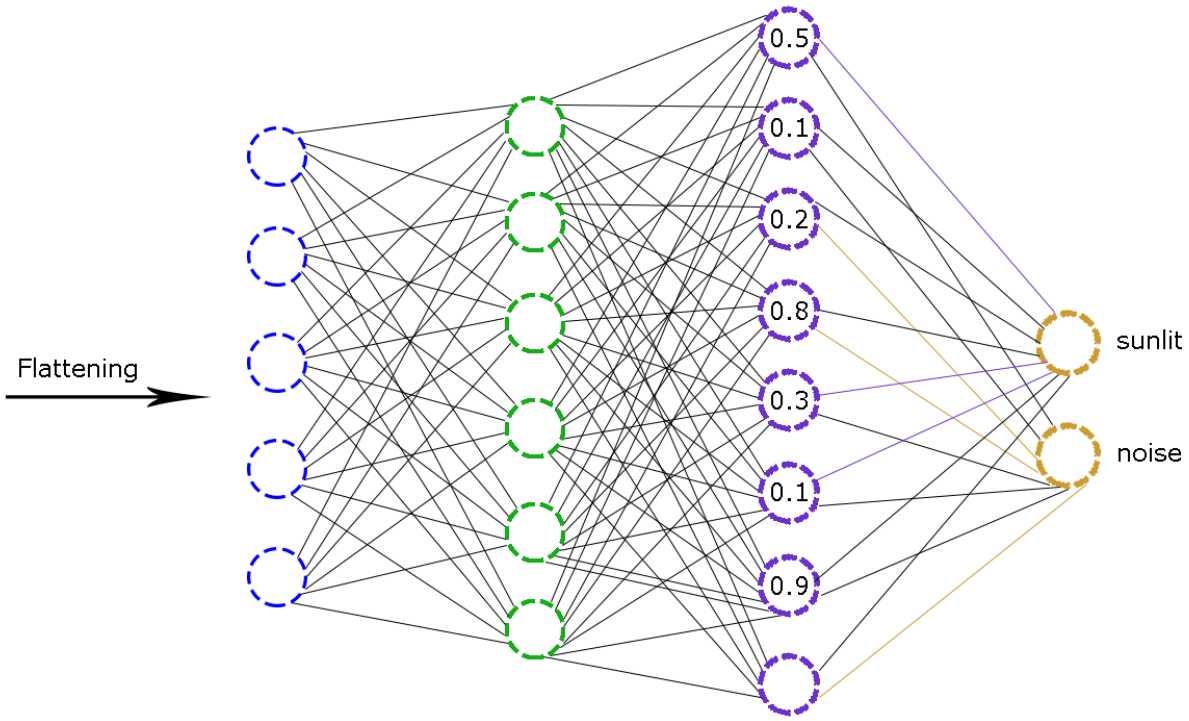


Figure 2.5 Fully Connected Layers in a CNN

classification label.

The fully-connected section of the CNN performs its own back-propagation to estimate the optimal weight values. The weights that reach each neuron prioritize the best candidate label. The final classification decision emerges after a voting process that takes place among the neurons.

2.2.3 Semantic Image Segmentation

Semantic image segmentation [30, 13, 37, 10, 56, 7] is a computer vision problem in which certain parts of an image are being classified, with respect to the image's visible information. In more detail, semantic image segmentation considers the annotation of each pixel with the representative class, as shown in Figure 2.6. It is note-worthy that there is no distinction between different occurrences of the same entity, since the model only considers the category

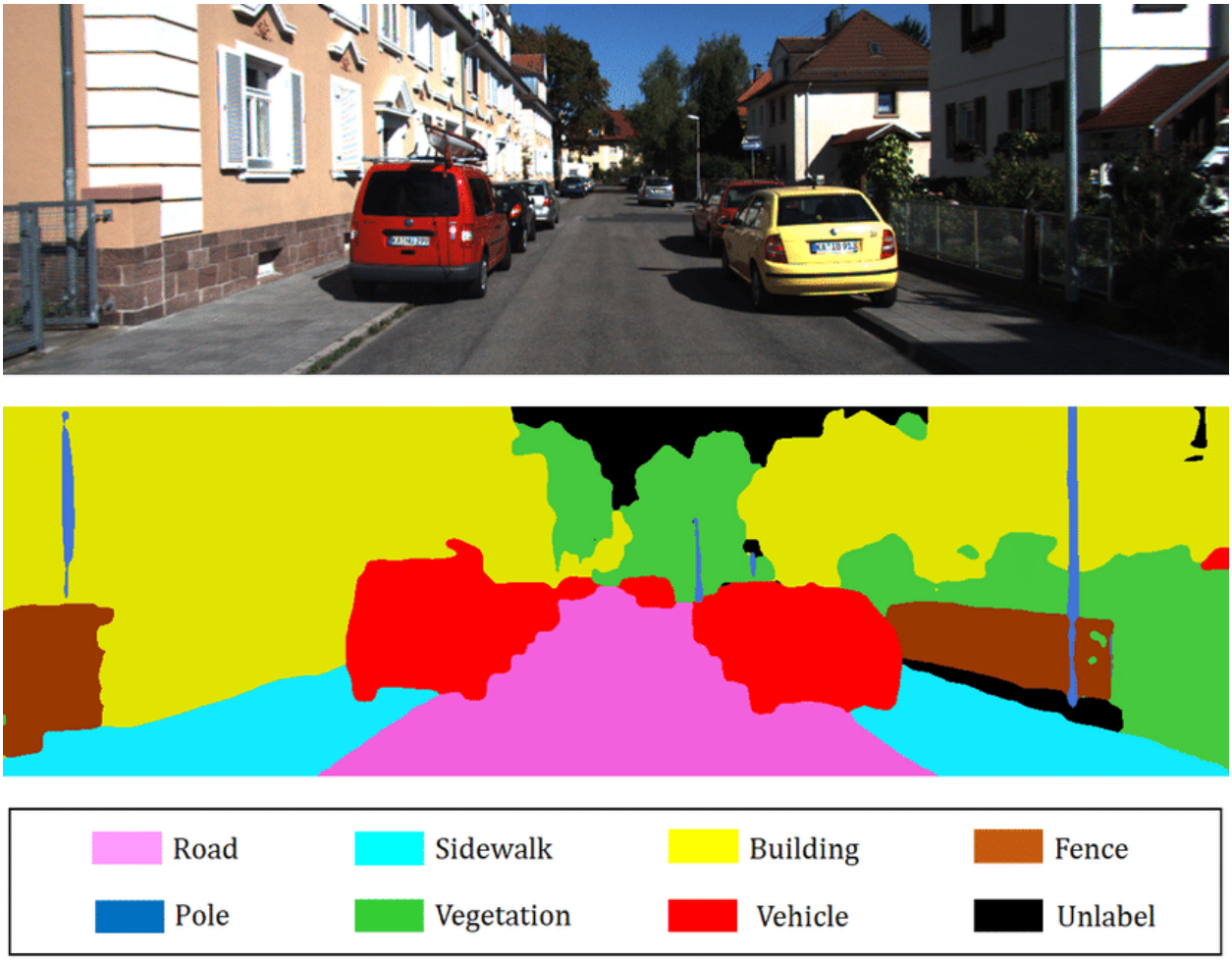


Figure 2.6 Semantic Image Segmentation

Source: [30]

a pixel belongs to. In case there are multiple occurrences of a given object in the image, the segmentation map does not identify these as separate objects, at least inherently. There is a different kind of models, called instance segmentation models, which identify distinct entities of the same class. Over the last years, the most promising semantic segmentation applications have been utilizing CNNs.

2.3 Web Development background

In this section we are going to discuss modern web development practices, the chosen technologies for each app component, and some of their advantages. Some basic concepts of the fundamental building blocks of web apps will also be explained, in an attempt to familiarize the reader with the process.

2.3.1 Microservices Vs. Monoliths

We designed the web app using microservices architecture. More and more web app developers choose the microservices approach, instead of the Monolithic one (Figure 2.7), at least in the context of large scale applications, since microservices offer much greater utility. On the downside, a microservices architecture costs a lot more to host. A microservices approach was by no means necessary for our app; however, it provided the opportunity to showcase modern web development methods.

Benefits of Microservices:

- Modularity
- Encapsulation
- Horizontally scaling
- Workload partitioning
- Automated operations
- On demand provisioning

2.3.2 Docker

Docker is emerging containerization technology for creating and managing Linux containers.

Benefits of Docker:

- Reproducibility

  A Docker container is Operating System (OS) agnostic. If Docker is installed, the

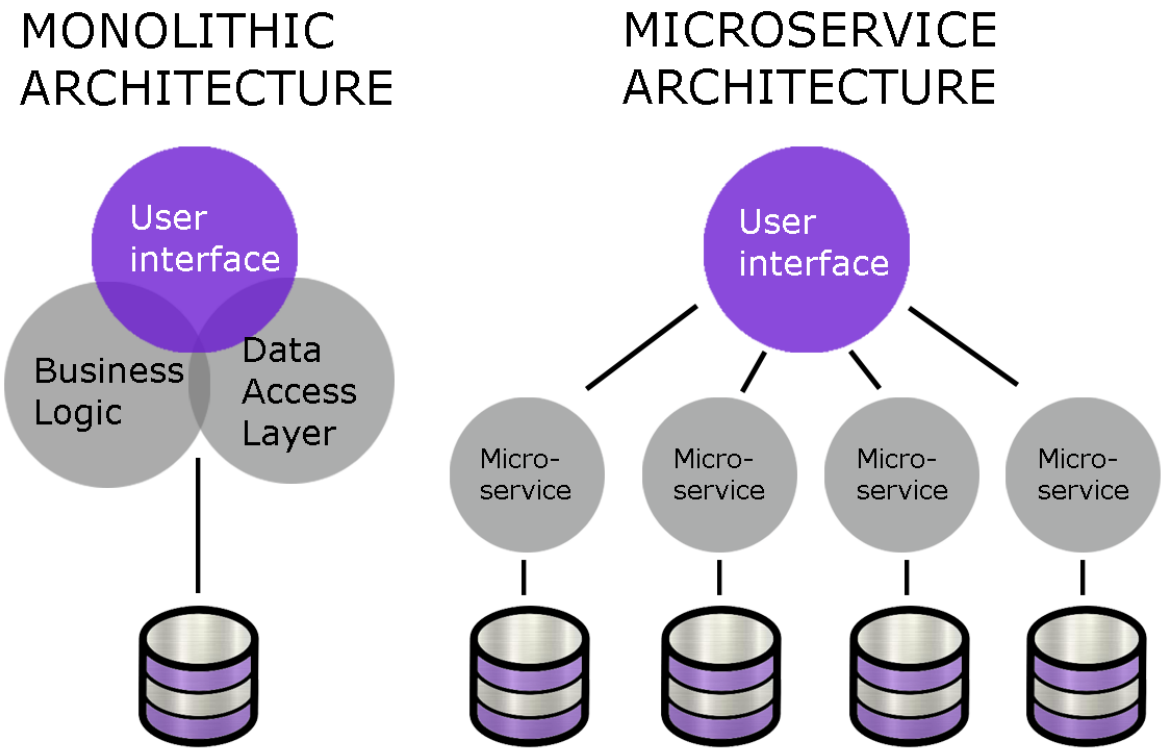


Figure 2.7 Monolithic vs Microservice architecture

container will run identically. A thorough description of the container configuration is stored inside that container's Dockerfile. As a result, having just the Dockerfile guarantees that the generated docker image for a container will be exactly the same across different builds. Changes on the configuration of the container can be managed through modification of this file, and are easy to document.

- Isolation

  All installed libraries for a given container affect that container only. By isolating the environment where each container operates, there are no dependency conflicts. A server can house multiple projects, each having a plethora of containers, with different versions of the same library or framework, without any issues.

- Security

  Building the different modules of a project into different containers has security advantages. For example, if the security of a container is breached, others containers are not affected.

- Docker Hub

  Similar to Github, Dockerhub serves as a storing, maintenance and distribution and version control platform for docker images. On Dockerhub, a developer can find information regarding the installation process, bug fixes and latest features of popular well-maintained docker images. Such images give developers the ability to kick-start their project, with pre-configured popular workflow stacks, such as LAMP stack (Linux, Apache, MySQL, PHP/Perl/Python).

- Continuous Integration

  Docker can be easily integrated into modern industry-standard production pipelines, with every update of a project automatically being stored as a new docker image and uploaded to Dockerhub before it is pushed to production.

Comparing Containers to Virtual Machines

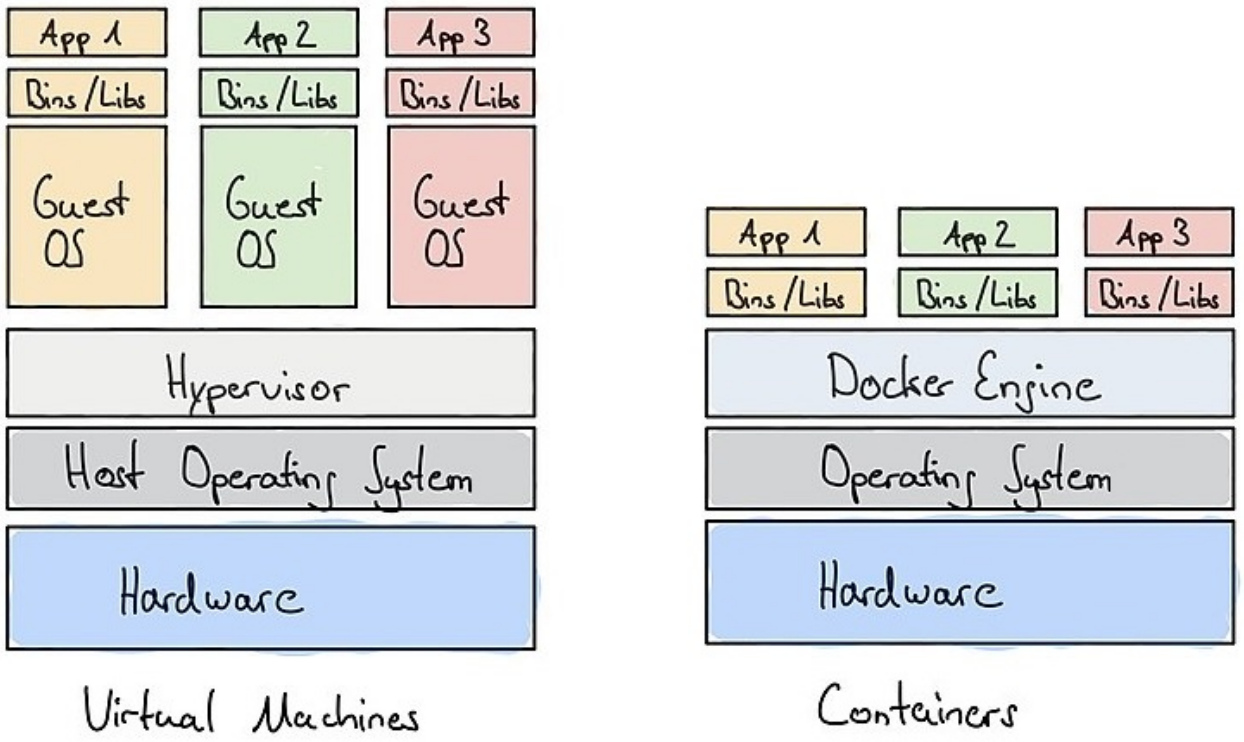


Figure 2.8 VMs vs Containers

Source: https://rb.gy/pcgmrb

Virtual machines (VMs) and containers (Figure 2.8) have many similarities in the context of resource allocation and isolation, but also fundamental differences. Containers virtualize the OS, while Vms virtualize the hardware directly.

Each **container** runs in a single process in user space, where dependencies are packed together, and leaves the rest of the environment unaffected. Containers share the operating system's kernel, and access a pre-defined portion of the machine's resources. They occupy less space than virtual machines, usually in the scale of megabytes.

Virtual Machines (VMs) are orchestrated by a hypervisor and virtualize the hardware directly. Each server can run multiple Vms, and be effectively converted into multiple servers. Each virtual machine has its own OS, dependencies, drivers, etc., thus it takes up multiple gigabytes of space in the ROM.

2.3.3 Client

The entirety of the client was built using React. React is an open-source JavaScript library for building component-based clients with highly customizable and responsive UIs. It is developed & maintained by Facebook and a broad group of individuals and companies.

Benefits of React:

- Declarative

  React gives developers the ability to pre-define views of the client for each state of the application, making debugging easy. Partial rendering of components, and asynchronous code batching guarantee high app performance and robustness.

- Component-Based

  Components can be abstractly grouped into stateful and stateless. Stateless components do not need to keep track of the current state of the application, and their functionality is not reliant on such information. Stateful components, on the other hand, need to keep track of the application's state and dynamically adjust their behavior. A developer can design these kind of modules from the ground up, starting with simple building blocks and scale the design to elaborate modules. Data can be directly passed from higher logic levels to low-level components.

- Plethora of Developer tools

  One of React's strong assets is rapid prototyping. This can be achieved using a wide range of modules/plug-ins that individuals or companies develop, test, document and maintain.

2.3.4 Server

For development a simple WSGI server was used, namely uWSGI, a project built for reliability, performance, versatility and low resource utilization.

Flask is a barebone WSGI web app framework, very popular among Python web developers. Flask provides all the essential tools needed to implement a back-end, with no excess modules. Developers can modify the framework to their needs by including external tools or libraries to the project.

2.3.5 Database

Regarding the database, we picked mongoDB, a document-oriented, distributed database built to embrace the cloud era. MongoDB stores documents in JSON format, making it ideal for storing the contents of an HTTP request or providing an HTTP response, with minimum pre-processing.

2.3.6 Reverse Proxy & File Server

A reverse proxy is a server that intercepts requests before they reach the origin server. This behavior resembles that of middleware. Reverse proxies are built in front of the actual server and act as an extra node, making sure no request reaches a particular server without passing through the proxy. Forward proxies, on the other hand, are built in front of clients and their role is to handle a server's response before it reaches the client (Figure 2.9).

Reverse proxies perform several operations after intercepting a request, and then communicate with the server. In return the server sends out a response to the reverse proxy, the response is manipulated, and then transferred to the client.

Serving static files, including user-uploaded files via flask is acceptable in development, but not in production, since it is extremely slow.

To satisfy the above needs, we mobilized Nginx, which acts both as a reverse proxy and a file server (and possibly in the future as a load balancer).

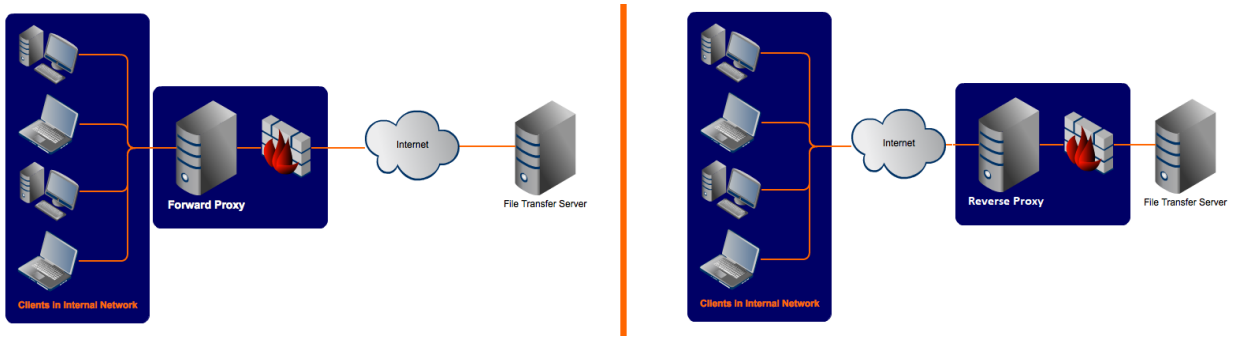


Figure 2.9 Forward proxy vs Reverse proxy

Source: https://rb.gy/mfuzqq

Benefits of a reverse proxy:

- Load balancing

  A high-traffic domain employs several different origin servers to handle the workload. The amount of load that each of these origin servers receives, and the order that dictates when each server should be employed is determined by reverse proxies. When a particular server gets over-loaded, incoming requests are re-routed to another server via the reverse proxy.

- Protection

  If a reverse proxy is built before the origin server, the latter's IP address is hidden to the public. Only the reverse proxy can communicate with the server. Aspiring, yet malicious individuals will only be able to attack the proxy, which will feature higher security protocols and employ more resources to encounter the attack.

- Global Server Load Balancing (GSLB)

  In simpler terms, load balancing that considers the geographic location of a client, and picks the closest server to handle that client's request.

- Caching

  Reverse proxies can store temporary response data and serve clients directly. In case the origin server is located in a different continent, but the reverse proxy is in the same

continent as the client, caching can drastically reduce communication time, and overall user experience

- Encryption

  Reverse proxies often alleviate the origin server from communication encryption, a labor-intensive and resource-demanding task.

Chapter Three

Our approach

In this chapter, we will discuss our semantic image segmentation method of choice and the baseline methods chosen from published literature. To evaluate the mentioned methods, we created our own dataset of sunlit Pistachio leaves. Regarding the dataset, we will elaborate on the data acquisition process, present the used equipment and highlight the annotation process.

3.1 Convolutional Neural Netowkrs (CNNs)

To implement semantic image segmentation, we mobilized two different CNN architectures, Full-Resolution Residual Networks, and DeepLabV3.

3.1.1 Full-Resolution Residual Network (FRRN)

Full-Resolution Residual Network (FRRN)[42] is a multi-scale processing technique. It is based on two streams. One stream goes through a sequence of pooling operations and is responsible for understanding large-scale relationships of image elements; the other stream maintains feature maps at the full image resolution, resulting in precise boundary adherence. This idea is visualized in Figure 3.1, where the two processing streams are shown in blue and red. The two streams are coupled using the Full Resolution Residual Units (FRRU).

The residual stream is computed by adding successive residuals, while the features on the other stream; the pooling stream, are the direct result of a sequence of convolution and pooling operations applied to the input. Since its training uses an end-to-end network to keep the two-stream connected, the FRRN performs joint processing of feature maps from both streams and combines their results after each max pooling. The FRRN has a mean IoU score of 71.8% when used with the Cityscapes dataset[10].

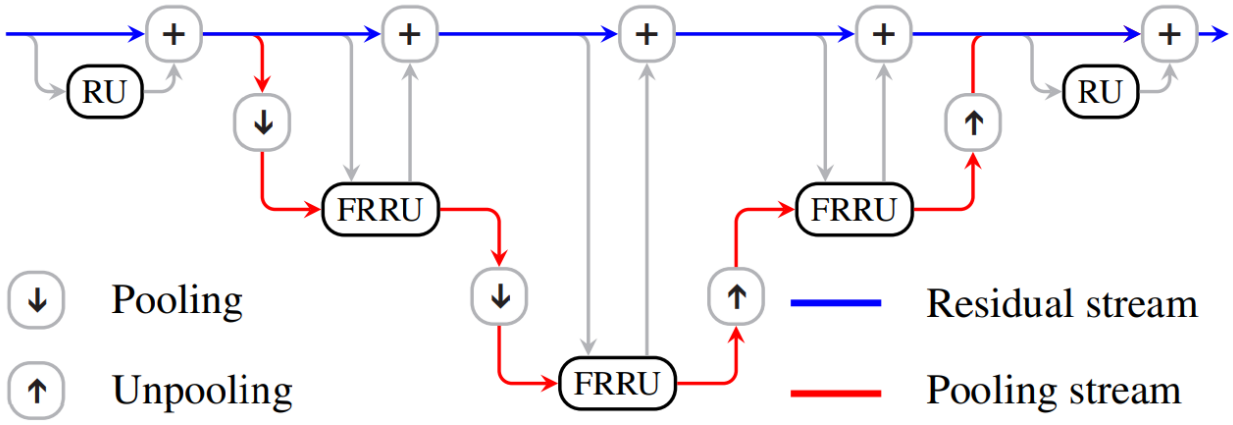


Figure 3.1 Full-Resolution Residual Neural Network

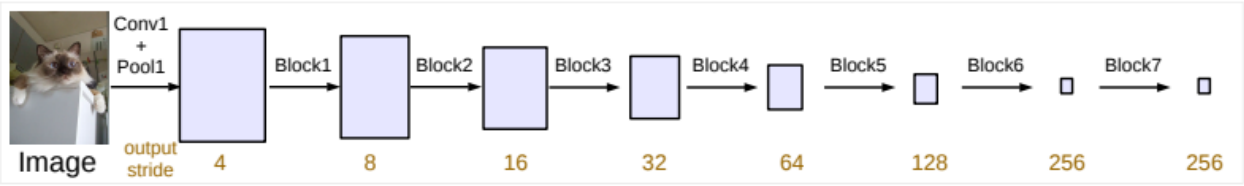
Source: [42]

3.1.2 DeepLabV3

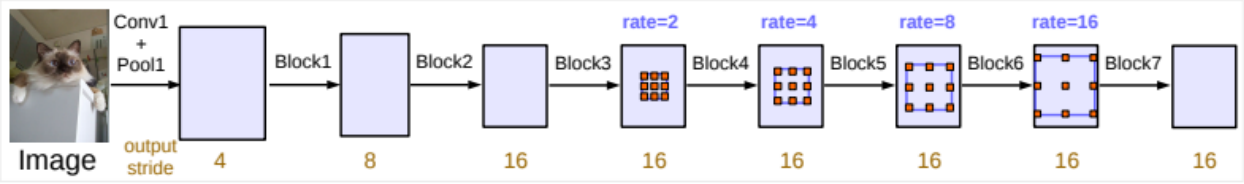
DeepLabV3[8] employs the atrous convolution[21, 15, 47, 41] with upsampled filters to extract dense feature maps and to capture long range information. In more detail, to encode multi-scale context, the cascaded module (Figure 3.2) gradually doubles the atrous rate while the atrous spatial pyramid pooling module (Figure 3.4) probes the features with filters at multiple sampling rates.

Standard convolution over feature map x, with the addition of atrous rate r:

$$y[i] = \sum_k x[i + r \cdot k]w[k] \tag{3.1}$$



(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

Figure 3.2 Cascaded modules without and with atrous convolution.

Source: [8]

Where the atrous rate r corresponds to the stride of the input signal sampling, which is equivalent to convolving the input with upsampled filters produced by introducing $r - 1$ zeroes between two consecutive filter values along each dimension. Standard convolution is a special use case for $r = 1$. Atrous convolution allows for adaptive modification of the filter's effective field-of-view by changing the r parameter. The effect of the atrous rate is illustrated in figure Figure 3.3.

If we use low atrous rate, low/low-scale information can be processed. If we increase the atrous rate, global/high-scale information can be processed. Thus, the DeepLabV3 model uses the atrous convolutions with different atrous rates to capture multi-scale information. DeepLabV3 has a mean IoU score of 81.3% when used with the Cityscapes dataset [10]. The ImageNet-pretrained [43] ResNet [20] that is used is ResNet101 which is found to perform better than ResNet50[8]. The pre-trained model has been trained on a subset of COCO train2017, on the 20 categories that are present in the Pascal VOC dataset and has global pixel wise accuracy of 92.7%.

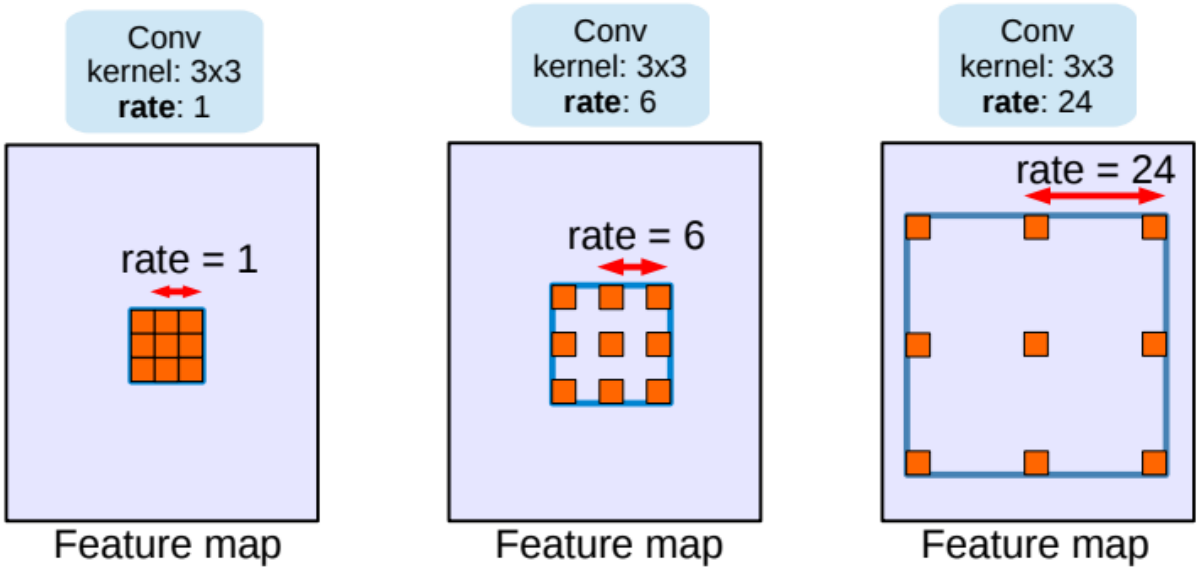


Figure 3.3 Atrous convolution with different stride rates in a 3x3 kernel.

Source: [8]

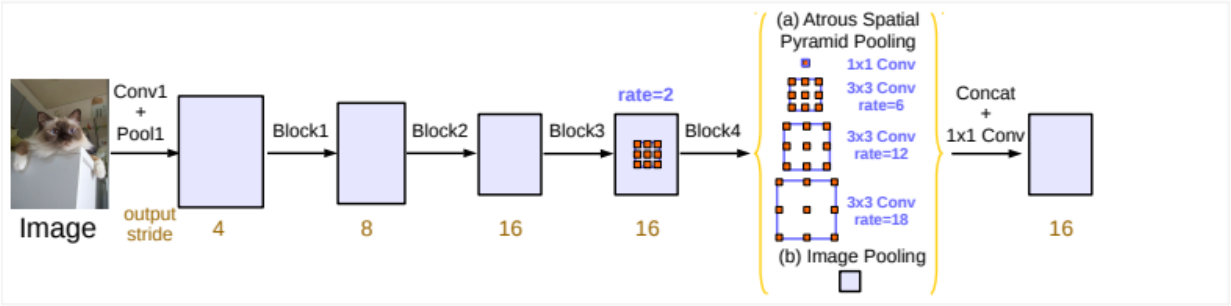


Figure 3.4 Parallel modules with atrous convolution (ASPP), augmented with image-level features.

Source: [8]

3.2 Sunlit Leaves identification baseline methods

In order to establish a comparison baseline for our CNN predictions, we implemented both Histogram Gradient Thresholding (HGT) and Gaussian Mixture Models (GMMs).

3.2.1 Histogram Gradient Thresholding (HGT)

Histogram gradient thresholding is a non-referenced method based on the research work of [45]. The research work also introduces two other methods to identify the canopy temperature, one of them is a referenced method and other is a non-referenced method, namely histogram thresholding at one or more standard deviation (SD) above and below the mean. As per the research work, histogram gradient thresholding has been found to be more accurate in defining canopy pixels and calculating canopy temperature from each thermal image (shaded and sunlit) when compared to the standard reference temperature thresholding method.

Histogram gradient thresholding was implemented in python and published under the following Github repository: https://github.com/rajeswarice/HistogramGradient. Using the temperature values obtained from the flir image metadata we constructed the temperature histogram by re-organizing the temperatures into equal interval class bins (one degree Celsius) and calculated the number of pixels as percentage of total number of pixels, for each bin. To avoid over-fitting, one-degree bin interval is considered. Observing Figure 3.5, the peak and main body of the histogram are the tree canopy pixels.

Our area of interest is the temperatures derived from the pixels of the tree canopy. Therefore, we need to remove the lower tail, which corresponds to sky pixels and the upper tail which represents pixels of non-leaf material. To determine the temperature limits, we must calculate the Ratio pixel change (RPC) lower and upper values which will, in order, give us the min and max temperatures. RPC is calculated as follows:

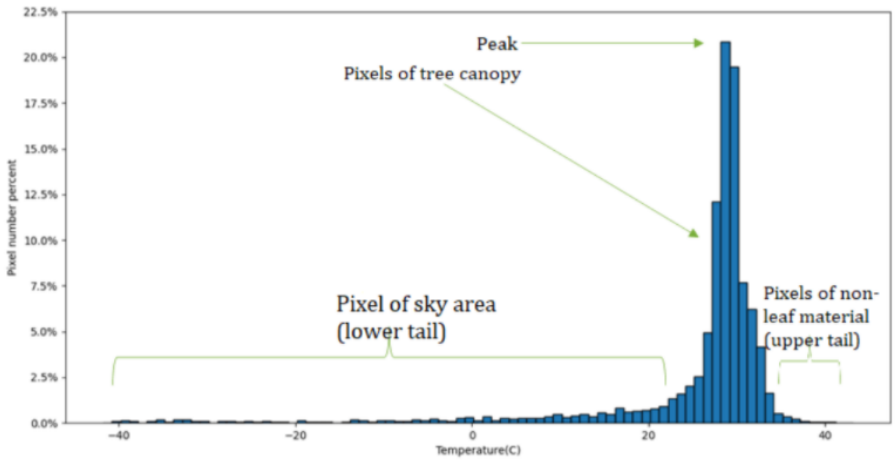


Figure 3.5 Histogram gradient thresholding

$$RPC = \frac{\Delta\, pixel\, number\, \%}{Class\, bin\, interval} \tag{3.2}$$

An iterative process is used to change the predefined gradient value symmetrically and incrementally from 0 %/C (corresponding to the flat portions of the histogram tails) to the maximum gradient. At each pair of RPC values, the value of Tc (Tc-predicted) is calculated as the population average and compared to the actual Tc value (Tc-actual) obtained from the manual annotation. Actual canopy temperature is calculated by finding the temperature from the underlying leaf area of the manually annotated images getting the average. In our case, we have used 10 images to calculate the actual average canopy temperature. The result we obtain after the above process is as shown in Figure 3.6. Unfortunately, the generated segmentation is insufficient to calculate the crop water stress, as it does not identify sunlit leaves clearly.

3.2.2 Gaussian Mixture Models (GMMs)

We used Gaussian mixture models as a color identification techinque, inspired by the research work of [53]. The paper introduces leaf identification based on the mean and variance of the RGB bands of visible spectrum images.

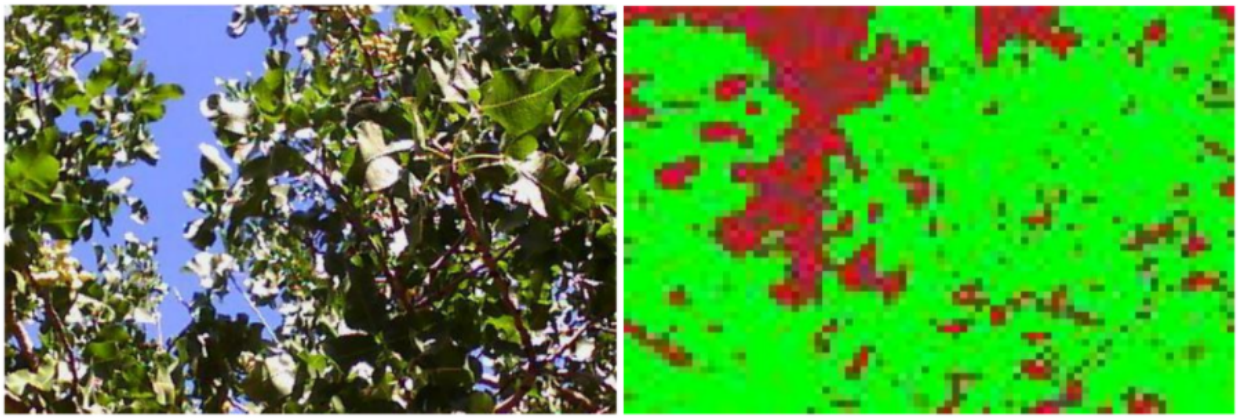


Figure 3.6 (left) RGB image, (right) Image segmented using histogram gradient thresholding

We, in turn, use the visible spectrum RGB image and the respective labeled image as input for the Expectation Maximization (EM) algorithm, explained in section 2.2.1, to find the parameters, namely the mean and covariance of the RGB color distribution in the sunlit areas of the image. When training begins, the EM parameters, are initialized randomly. The calculated parameters are stored and used to initialize the model and predict the segmentation masks for the test set. Figure 3.7 shows the result of the image segmentation using the Gaussian mixture model approach.

The Gaussian mixture model yields a better result detecting the sunlit leaves compared to histogram gradient thresholding. Despite being better, a drawback of this approach is that not everything that is green or yellow color-wise is sunlit leaves. Our crop of focus is pistachio, which is found in multiple shades of yellow. The Gaussian mixture model falsely identifies pistachios as sunlit leaves. Thus, this technique is not the best approach to identify sunlit leaves.

Gaussian mixture models were implemented in Python and published under the following Github repository: https://github.com/rajeswarice/GMM.

Figure 3.7 (left) RGB image, (right) Image segmented using gaussian mixture models

3.3 Dataset

Our dataset consists of 1,432 images of pistachio trees in the visible and infrared spectrum with their corresponding labels for sunlit leaf semantic segmentation. The complete process can be seen in [Figure 3.8]. The images were collected during the summer of 2020 under clear sky on sunny days. The images also contain weather meta-data encoded by the FLIR camera. The dataset is released on kaggle under the following url: https://www.kaggle.com/datasets. Sample images of the dataset are provided in [Figure 3.11, Figure 3.12, Figure 3.13]

3.3.1 Data acquisition

To capture images for our dataset we used FLIR AX8 thermal imaging cameras (Figure 3.9) mounted on a custom made camera stand(Figure 3.10) for uniformity, stability and ease of use. These cameras provide thermal images of 640x480 resolution (Figure 3.11), that are compatible with the FLIR Tools+ (proprietary) software. These thermal images consist mostly of virtually generated pixels, since the true resolution of the thermal cameras, hence of useful information is just 60x80. The metadata contain a plethora of useful information (Table 3.1), such as weather data, material data, calibration constants, and the original visible spectrum image (640x480) (Figure 3.12) along with the raw thermal image (60x80)

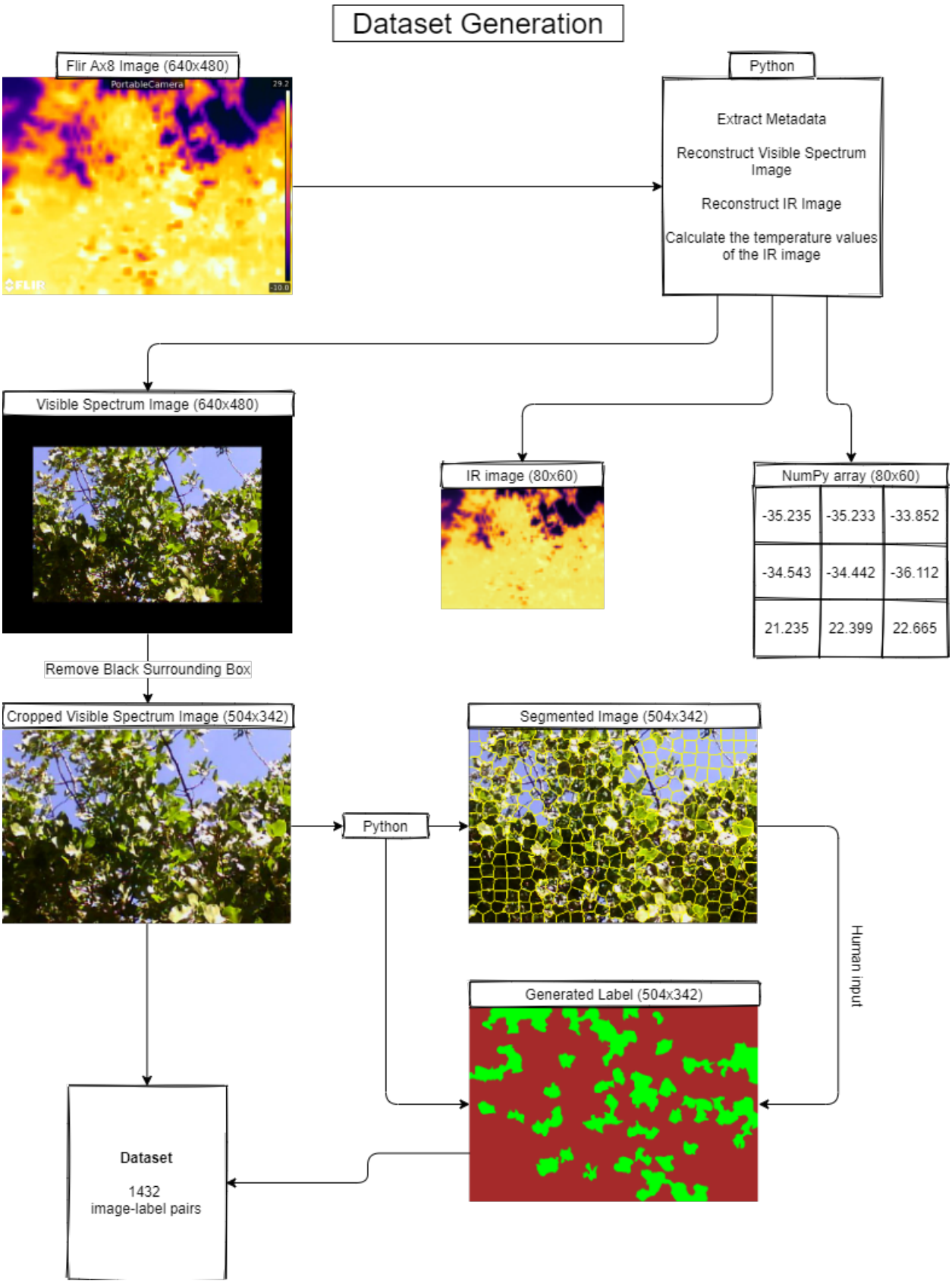


Figure 3.8 Dataset generation pipeline

in the far infrared region of the spectrum.

Since our goal was to also develop a web app, we had to reverse engineer some of the FLIR Tools+ functionality. To be specific, we developed a script that reads the image, extracts the metadata using EXIFtool and reconstructs the visible spectrum image, the thermal image, and the temperature array. This script along with various other utility scripts can be found at https://github.com/Citywalk3r/thermal_data_modifier_windows_version. It is worth noting that the original formula that calculates the temperature out of the parameters found in the Flir image can be found at https://github.com/gtatters/Thermimage/blob/master/R/raw2temp.R.

3.3.2 Annotation Tool

To annotate our images, we used a customized version of the open source software Pynovisao[46], a collection of tools, written in Python, easy to use via a friendly user interface. As described in the official release repository, this collection of tools allows the user to select an image (or folder) and realize numerous actions such as:

- Generate new Datasets and classes
- Segmentation of images
- Extract features from an image
- Extract frames from videos
- Train Machine Learning algorithms
- Classify using CNNs
- Experiment with data using Keras
- Create XML files from segments previously created

Regarding segmentation, the tool provides 3 unsupervised clustering algorithms (Slic[1],

Flir image metadata	
Weather-related data	Relative Humidity
	Atmospheric Temperature
Material Parameters	Emissivity
	Reflected Temperature
Encoded image data	Embedded Image (Binary)
	Raw Thermal Image (Binary)
Other	Object Distance
	IR Window Temperature
	IR Window Transmission
	Planck R1
	Planck R2
	Planck B
	Planck F
	Planck O

Table 3.1 Flir Image Metadata

Figure 3.9 Flir AX8 thermal imaging camera

Source: https://rb.gy/z2ufxv

Figure 3.10 Custom made camera mount.

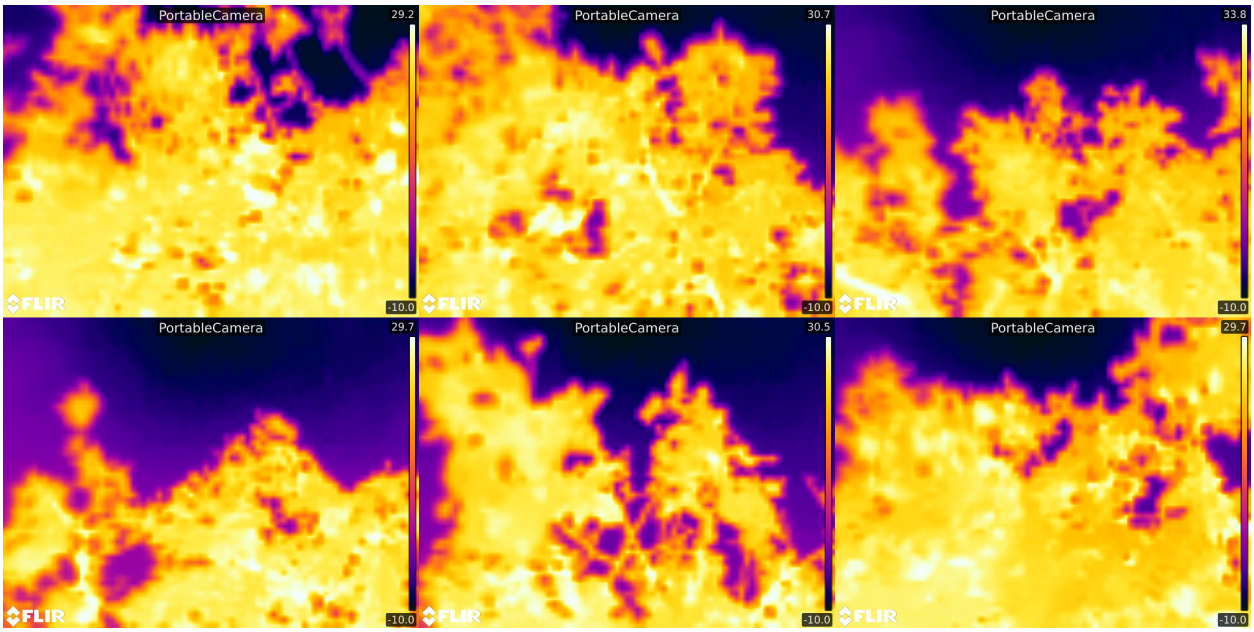


Figure 3.11 Thermal images captured with FLIR AX8

Figure 3.12 Visible spectrum images encoded in the metadata of the thermal images captured with FLIR AX8

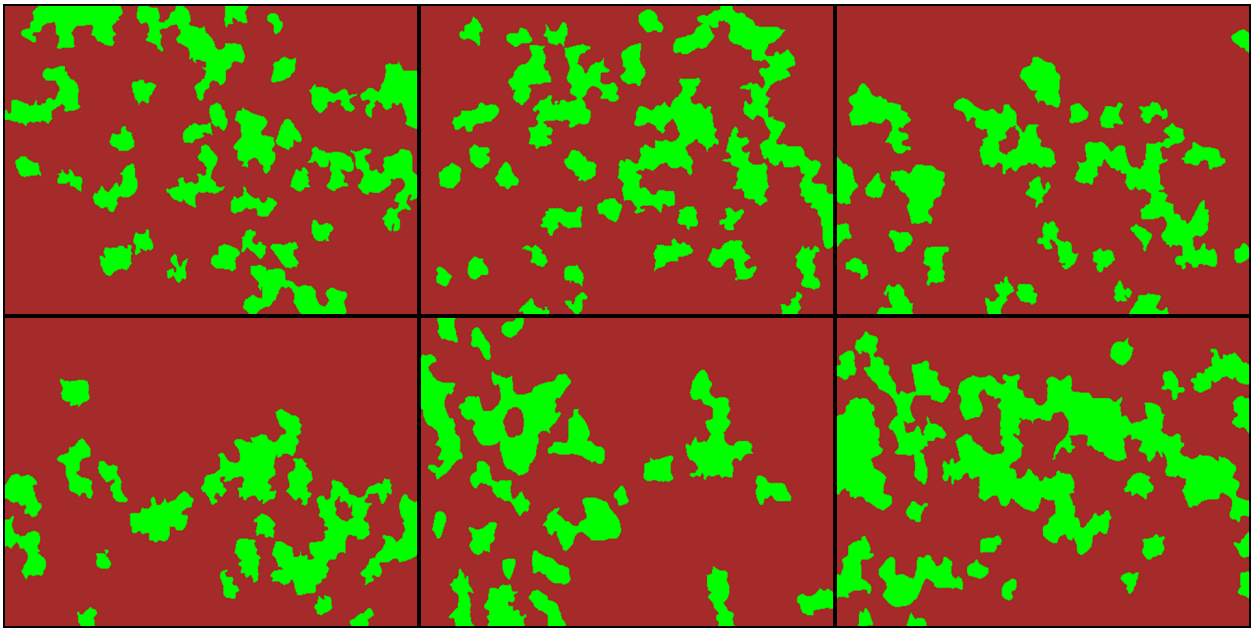


Figure 3.13 Human-generated labels.

Felzenszwalb[14] and Quickshift[44]). We decided that Slic best serves our needs, so we proceeded with it.

We classified each segment into two classes, Sunlit and Noise. The labeling process starts with the user opening an image via the menu. When that happens, a label image (with all pixels assigned to Noise) is created in a different folder. Then, the user executes the clustering algorithm. In Figure 3.14 you can see what a visible spectrum image looks like after Slic has been executed on it.

After the segmentation is performed, the user can click on the formed segments. When a segment is clicked, all the pixels surrounded by the segment's bounds are assigned the selected class. After all the segments corresponding to sunlit leaves have been identified and clicked, the generated label image is complete, as seen in Figure 3.15. Due to the fact that segmentation is not pixel perfect, we established that if a segment is more than 70% sunlit, the whole segment should be classified as sunlit.This process was performed for all 1432 images.

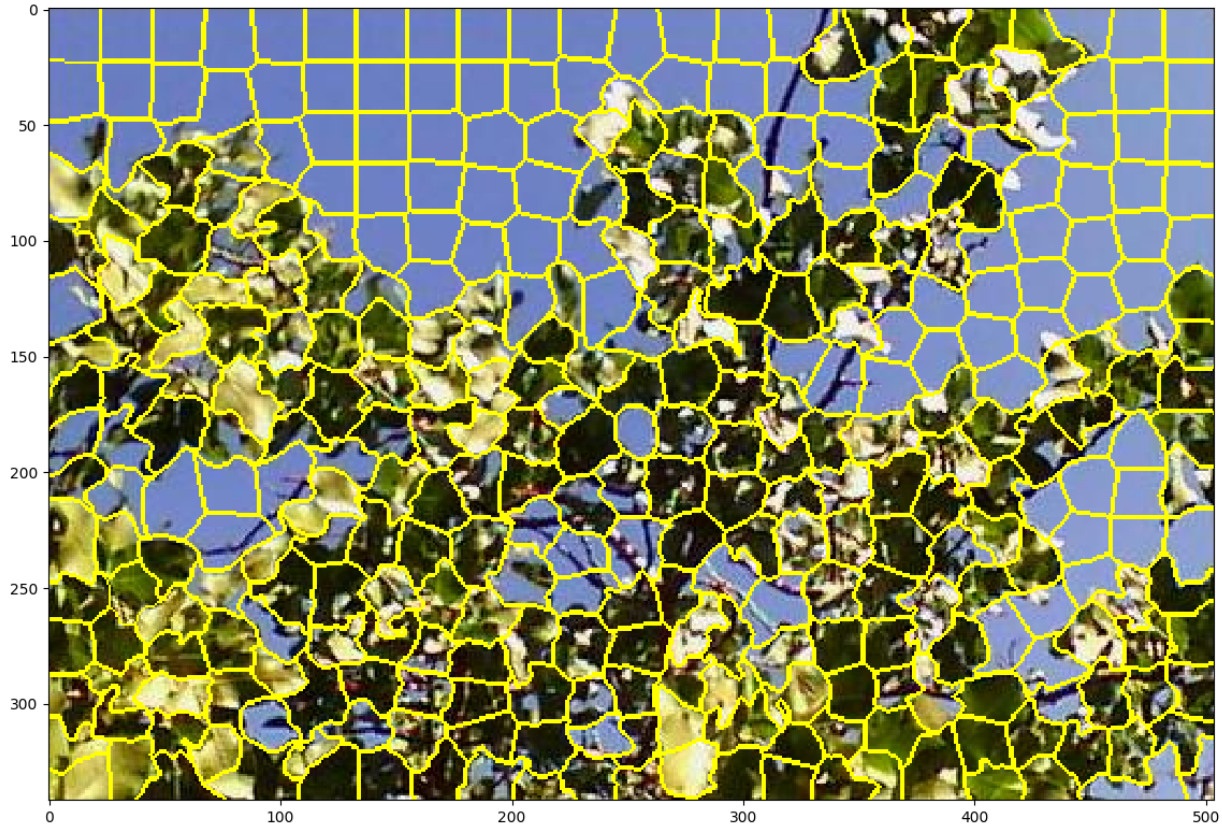


Figure 3.14 Image segmented with Slic

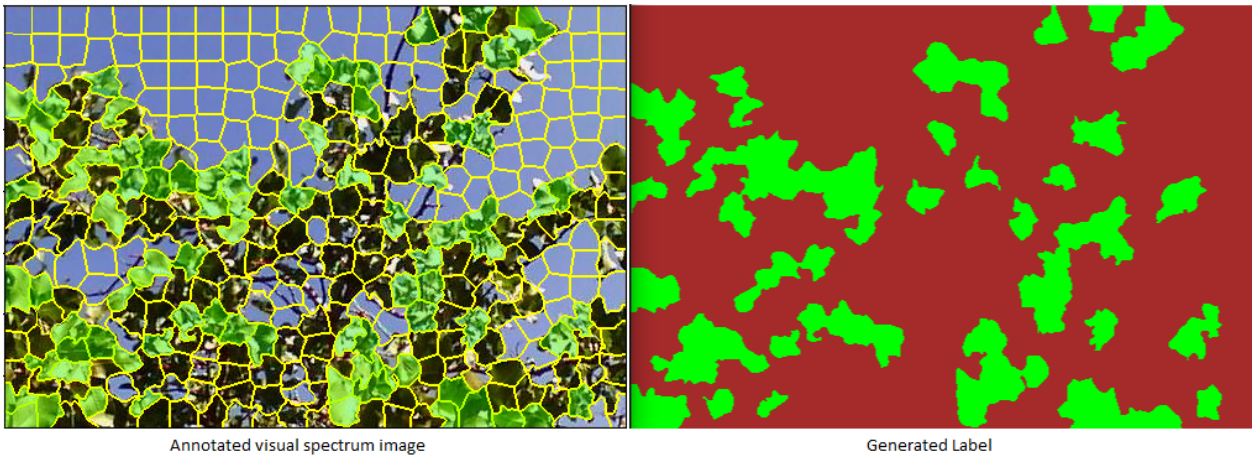


Figure 3.15 Annotated image and generated label

Chapter Four

Results

In this chapter we will discuss the results of the image segmentation techniques used in this work, namely a temperature method, histogram gradient thresholding (HGT), a color identification method, Gaussian Mixture Model (GMM), and 2 Convolutional Neural Network methods, Full-Resolution Residual Network (FRRN) and DeepLabV3. The performance is measured in terms of Hamming Distance, Pixel Accuracy, and pixel Intersection-over-Union (IoU). CNNs' training data and results are collectively presented in section 4.4.

Figure 4.1 and Figure 4.2 illustrate a compilation of the predictions of the selected methods, along with the respective overlays on top of the visible spectrum image. The HGT method gives the whole canopy, while the GMM finds most pixels of green-yellow color. The CNN predictions are much closer to the ground truth, with the FRRN model successfully excluding pistachio fruit from the sunlit leaves class, in spite of the color resemblance.

4.1 Hamming Distance

The Hamming distance between 1-D arrays u and v, is simply the proportion of disagreeing components in u and v. It measures the minimum number of substitutions required to change one string/vector into the other. Figure 4.3 shows a plot of the hamming distance obtained from the utilized methods. Results are normalized between 0 and 1. DeepLabV3 and FRRN

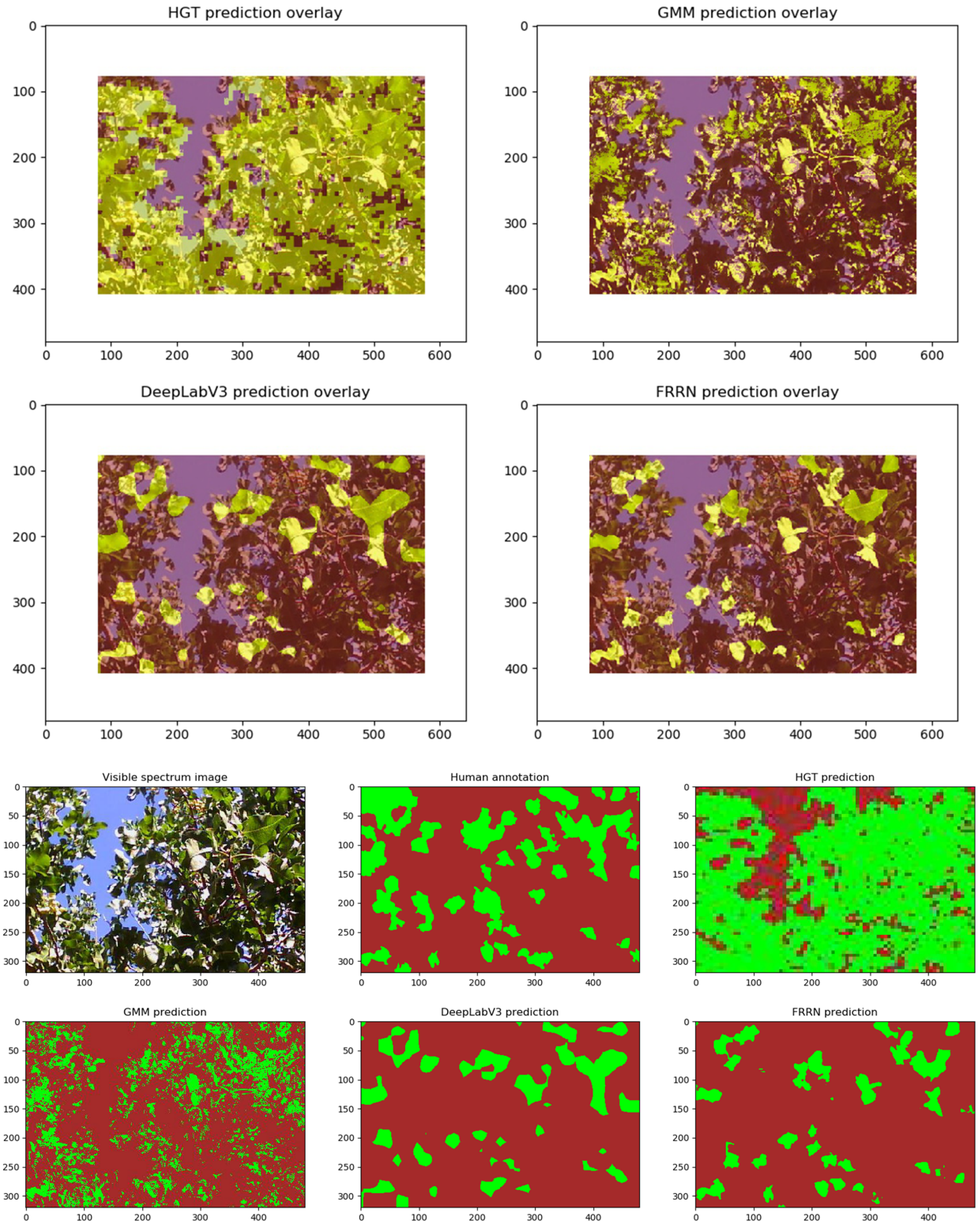


Figure 4.1 Predictions of tested methods overlaid on top of the visible spectrum image.

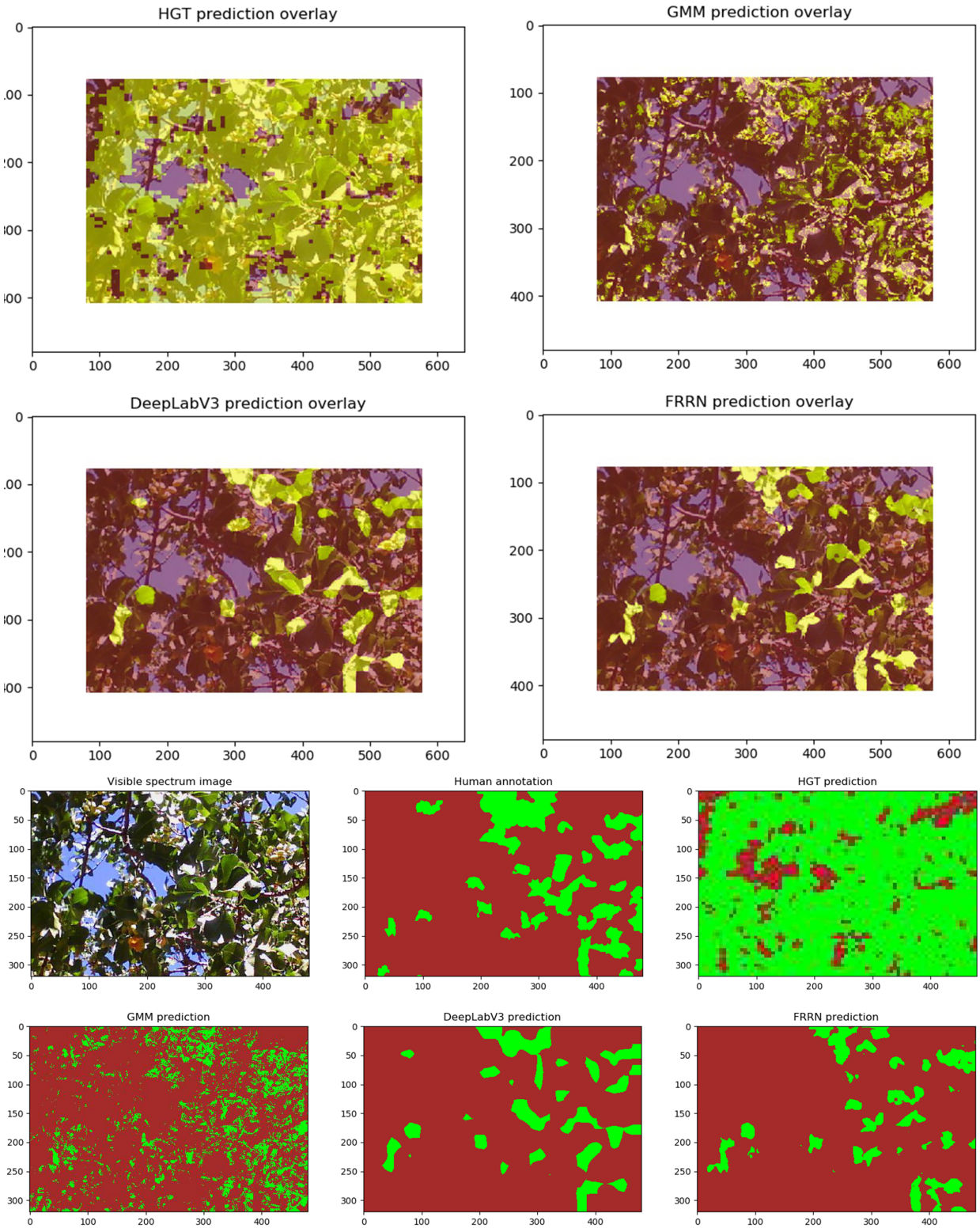


Figure 4.2 Predictions of tested methods overlaid on top of the visible spectrum image.

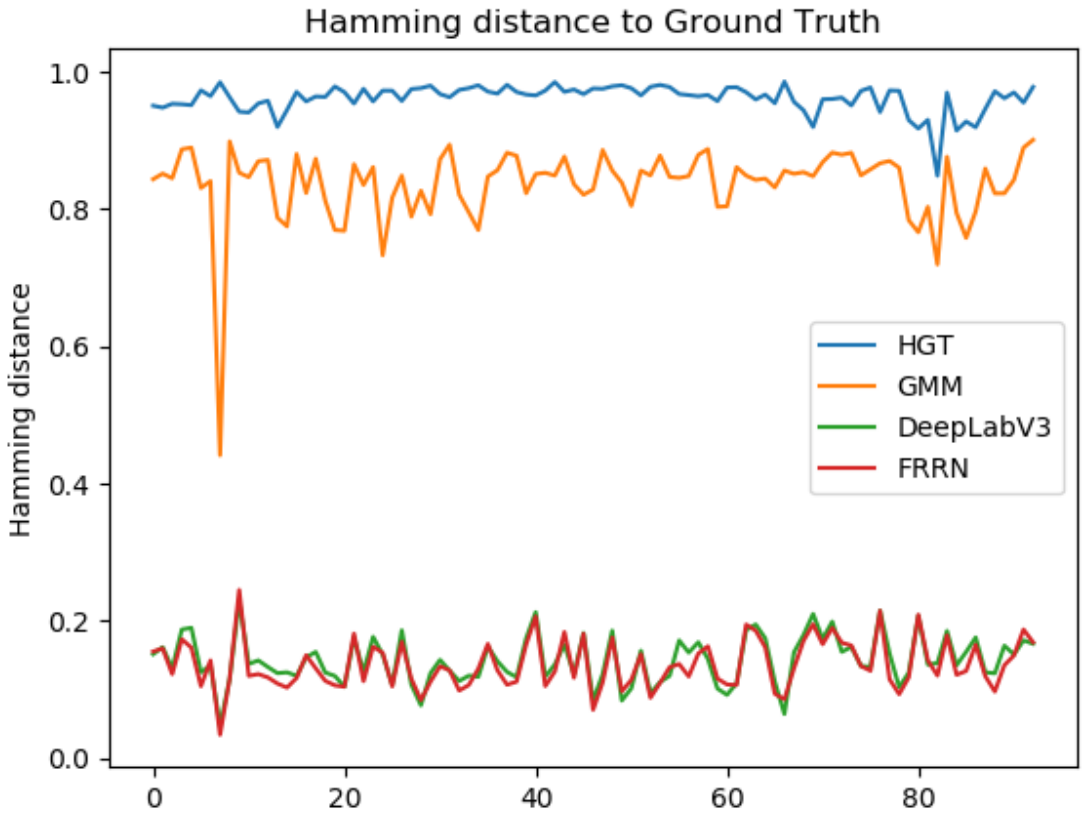


Figure 4.3 Hamming Distance of tested methods, lower is better.

achieve a low Hamming Distance of around 0.18, while Historgram Gradient Thresholding produces the highest.

4.2 Intersection over Union (IoU)

The Intersection over Union (IoU) metric, also referred to as the Jaccard index, is essentially a method to quantify the percent overlap between the target mask and our prediction output. It is calculated by the below formula

$$IoU = \frac{target \cap prediction}{target \cup prediction} \tag{4.1}$$

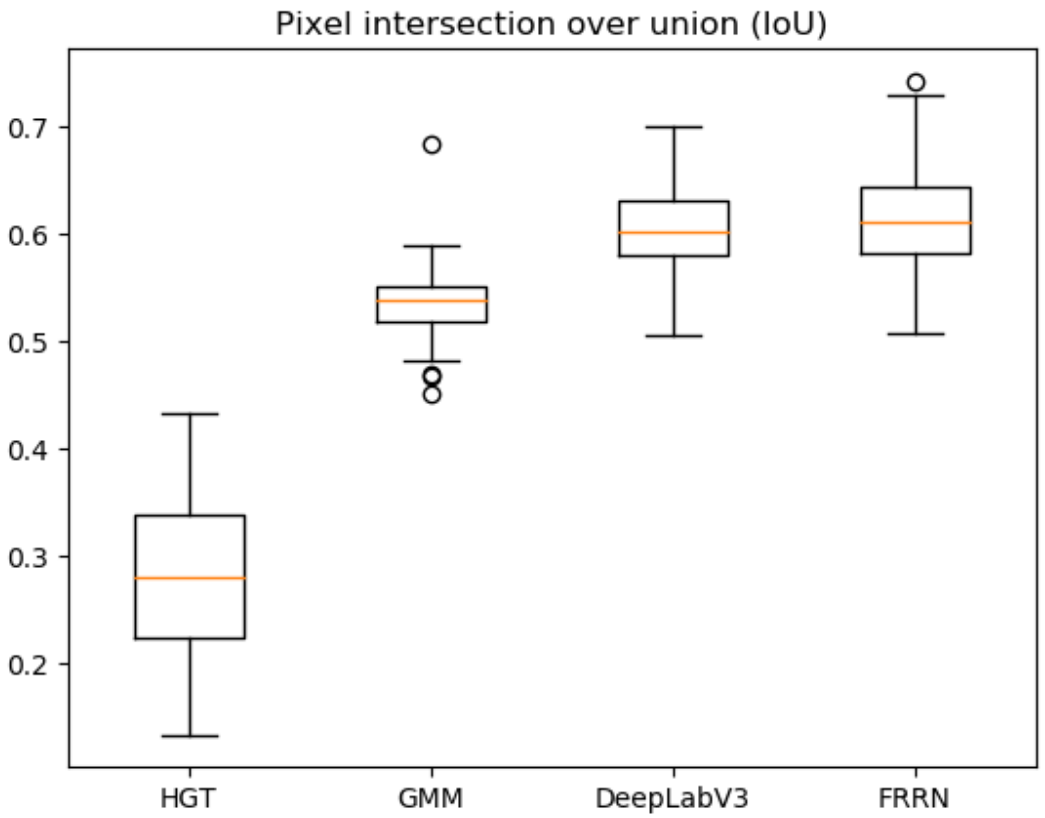


Figure 4.4 Pixel Intersection over Union (IoU) of tested methods, higher is better.

If the prediction is completely correct, IoU = 1. The lower the IoU, the worse the prediction result. Figure 4.4 shows a box plot of the IoU scores obtained from the different methods. From the box plot we can see that histogram gradient thresholding achieves the lowest mean IoU score of 0.28 and both CNN models, FRRN and DeepLabV3 have the highest meanIoU score of 0.62 and 0.61, respectively, as shown in Table 4.7 and Table 4.9. This indicates that DeepLabV3 and FRRN have identified sunlit leaves more accurately.

4.3 Pixel accuracy

Pixel accuracy will give us the percent of pixels in the image which are correctly classified.

		Prediction	
		Sunlit Leaves	Noise
Ground Truth	Sunlit Leaves	TP	FN
	Noise	FP	TN

Table 4.1 Pixel accuracy reference confusion matrix.

		Prediction	
		Sunlit Leaves	Noise
Ground Truth	Sunlit Leaves	30.67%	9.74%
	Noise	3.63%	55.96%

Table 4.2 Normalized mean pixel accuracy confusion matrix of FRRN.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

Where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative, determined as shown in Table 4.1. Figure 4.5 shows that histogram gradient thresholding has the least pixel accuracy of 45.12% compared to the other models. Gaussian mixture model segmentation has a pixel accuracy of 80.2%. Both DeepLabV3 and FRRN models have similar pixel accuracy of 85.76% and 86.63% respectively, as shown in Table 4.7 and Table 4.9. This indicates that CNNs have identified the sunlit leaves with the most accuracy.

		Prediction	
		Sunlit Leaves	Noise
Ground Truth	Sunlit Leaves	29.24%	9.11%
	Noise	5.13%	56.52%

Table 4.3 Normalized mean pixel accuracy confusion matrix of DeepLabV3.

		Prediction	
		Sunlit Leaves	Noise
Ground Truth	Sunlit Leaves	31.04%	2.22%
	Noise	52.66%	14.08%

Table 4.4 Normalized mean pixel accuracy confusion matrix of HGT.

		Prediction	
		Sunlit Leaves	Noise
Ground Truth	Sunlit Leaves	23.53%	16.5%
	Noise	3.3%	56.67%

Table 4.5 Normalized mean pixel accuracy confusion matrix of GMMs.

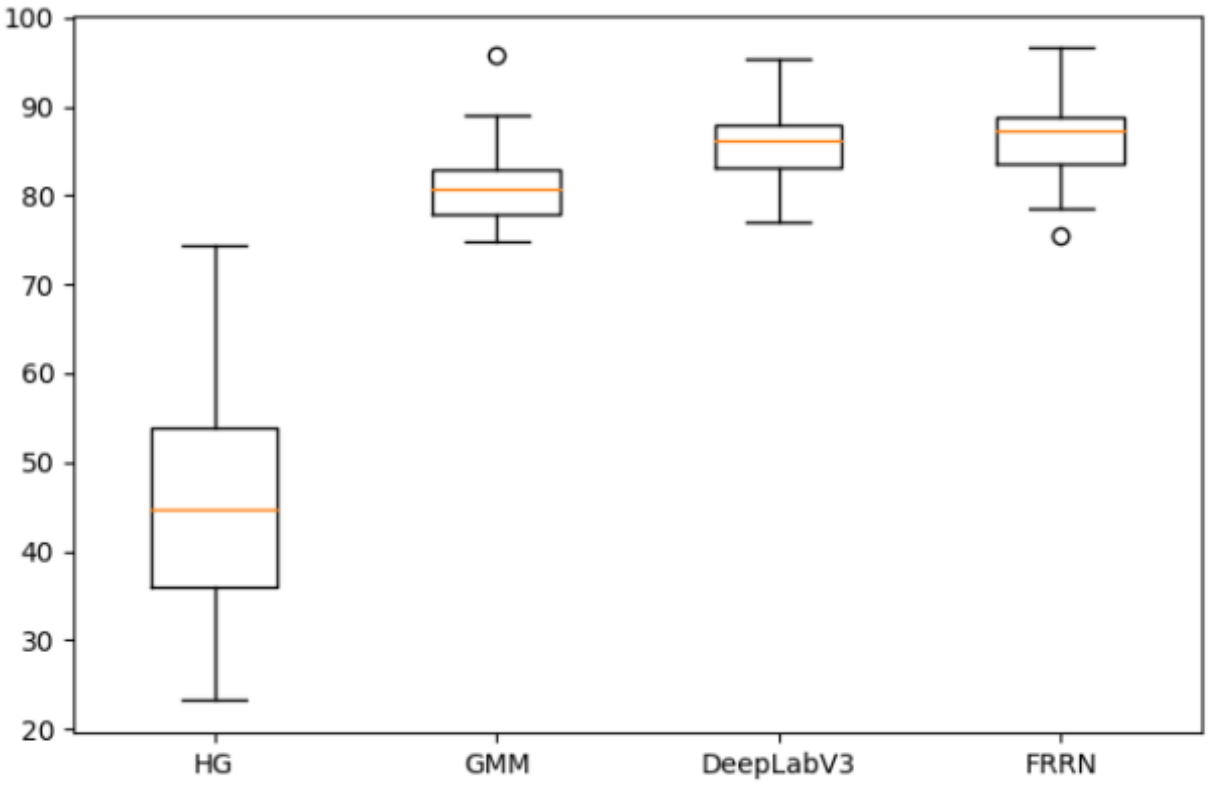


Figure 4.5 Pixel Accuracy of the tested methods.

4.4 CNN training complete results

We conducted several training trials on both FRRN and DeepLabV3, in order to obtain the best results possible. Below we highlight some training trials, monitor the training process, and present the results of the trained models on the test set:

4.4.1 DeepLabV3 results

Experiment 1: We used a training set of 592 images, 159 images as the validation set and tested the trained network on 93 images.The epochs were set to 150. The training process with the performance on the validation set is reflected in Figure 4.6. Table 4.6 shows the results of the trained network on the test set. These results are also illustrated in Figure 4.7.

Experiment 2: We used a training set of 592 images, 159 images as the validation set and tested the trained network on 93 images. The epochs were set to 350. The training process with the performance on the validation set is reflected in Figure 4.8. Table 4.7 shows the results of the trained network on the test set. These results are also illustrated in Figure 4.9.

4.4.2 FRRN results

Experiment 1: We used a training set of 592 images, 159 images as the validation set and tested the trained network on 93 images.The epochs were set to 150. The training process with the performance on the validation set is reflected in Figure 4.10. Table 4.8 shows the results of the trained network on the test set. These results are also illustrated in Figure 4.11.

Experiment 2: We used a training set of 592 images, 159 images as the validation set and tested the trained network on 93 images. The epochs were set to 300. The training process with the performance on the validation set is reflected in Figure 4.12. Table 4.9 shows the results of the trained network on the test set. These results are also illustrated in Figure 4.13.

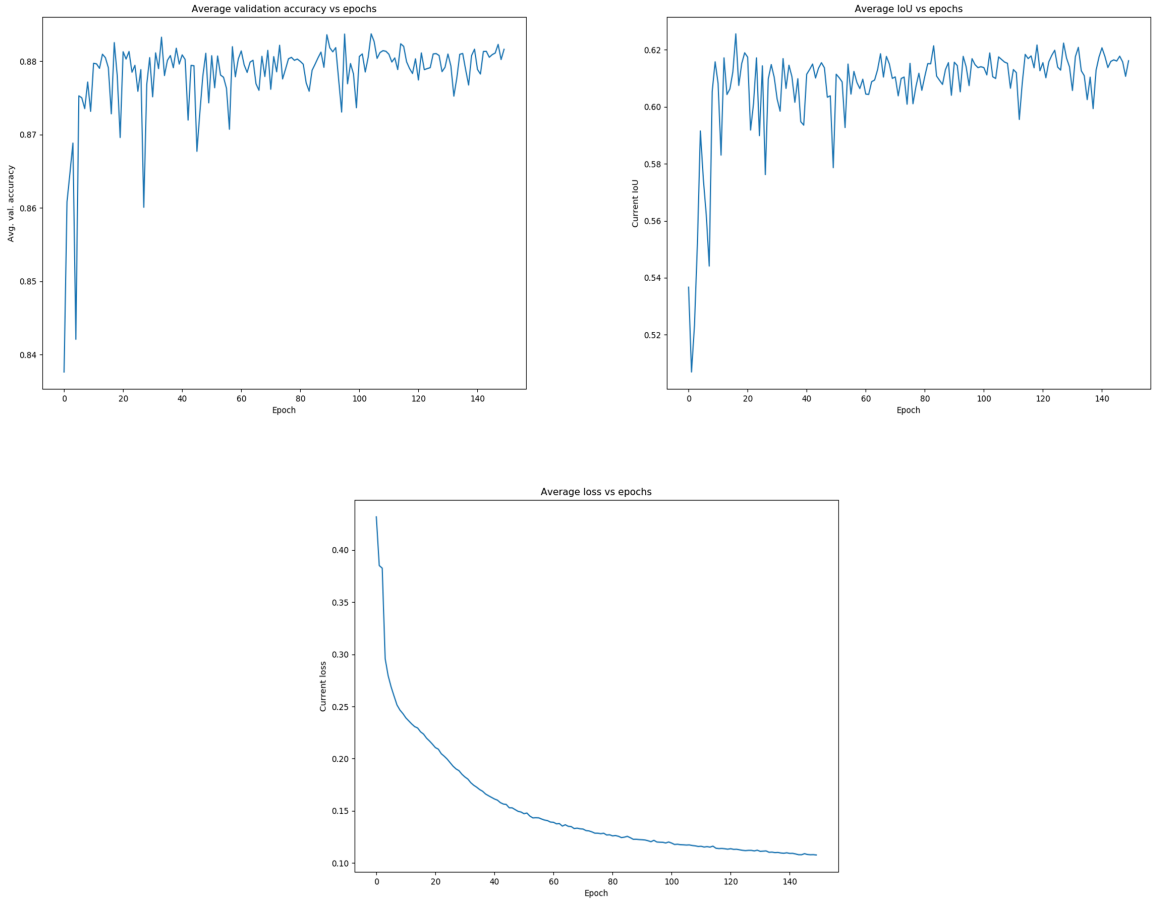


Figure 4.6 DeepLabV3 150 epochs training monitoring. Top Left: Avarage validation accuracy vs epochs, Top Right: Average IoU vs epochs, Bottom: Average loss vs epochs.

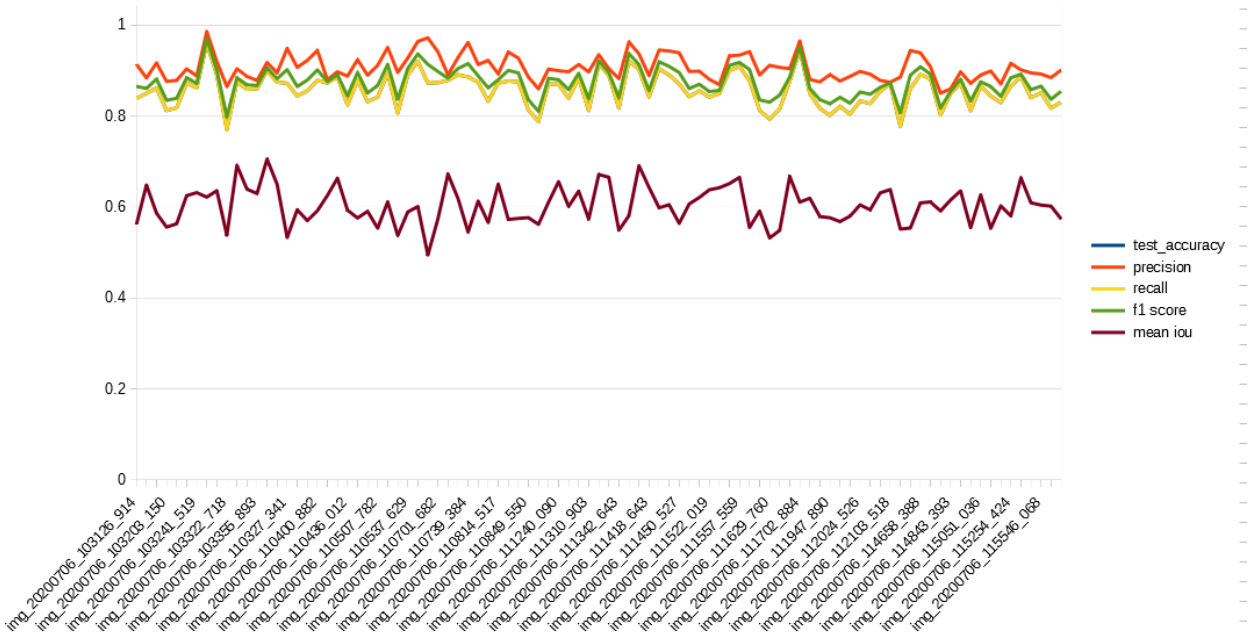


Figure 4.7 DeepLabV3 150 epochs results on the test set.

DeepLabV3 performance on the test set	
Training Set	592
Validation Set	159
Test Set	93
No. of epochs	150
No. of val. images/epoch	25
Accuracy	0.857599290322581
Precision	0.907120612903226
Recall	0.857599290322581
F1 score	0.874592784946237
IoU	0.602642096774193

Table 4.6 DeepLabV3 Experiment 1.

DeepLabV3 performance on the test set	
Training Set	592
Validation Set	159
Test Set	93
No. of epochs	350
No. of val. images/epoch	40
Accuracy	0.857404827956989
Precision	0.906433655913978
Recall	0.857404827956989
F1 score	0.874216580645162
IoU	0.602772021505376

Table 4.7 DeepLabV3 Experiment 2.

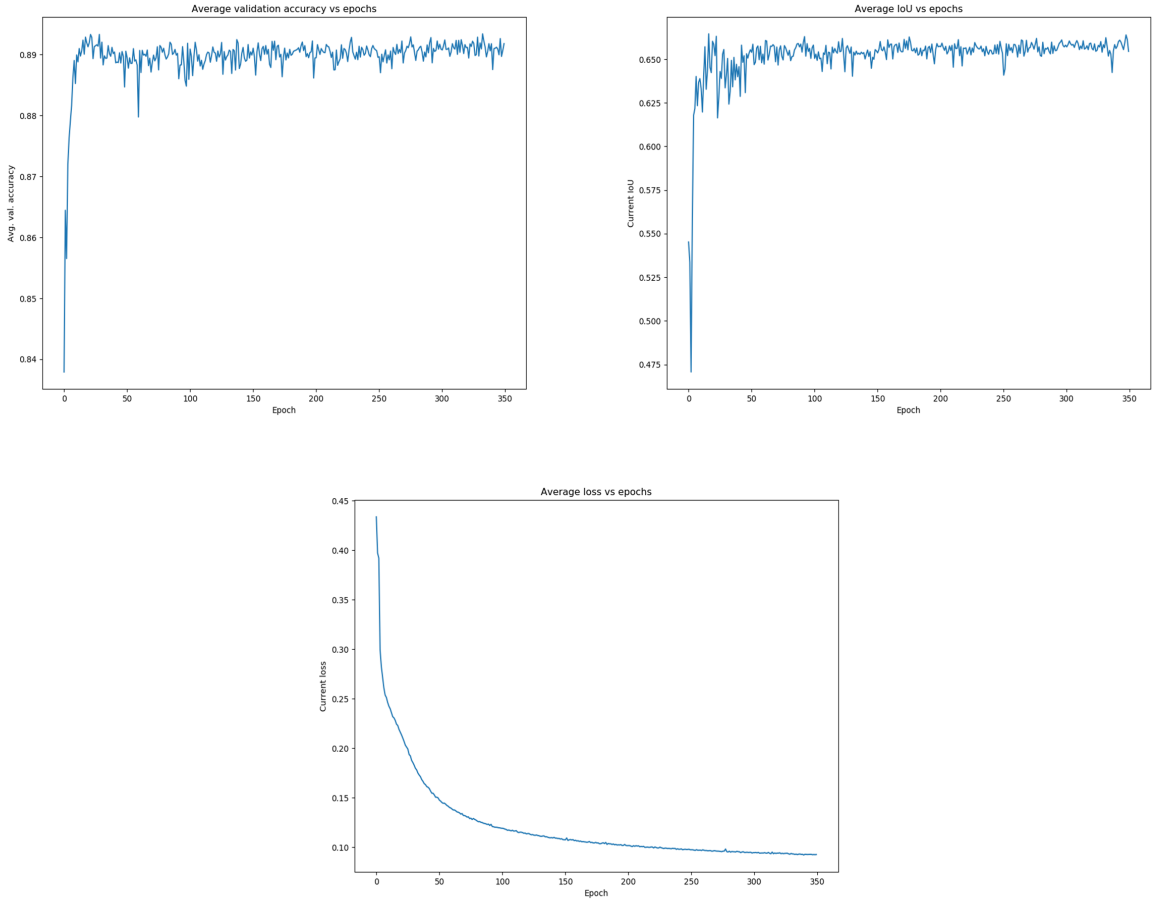


Figure 4.8 DeepLabV3 350 epochs training monitoring. Top Left: Avarage validation accuracy vs epochs, Top Right: Average IoU vs epochs, Bottom: Average loss vs epochs.

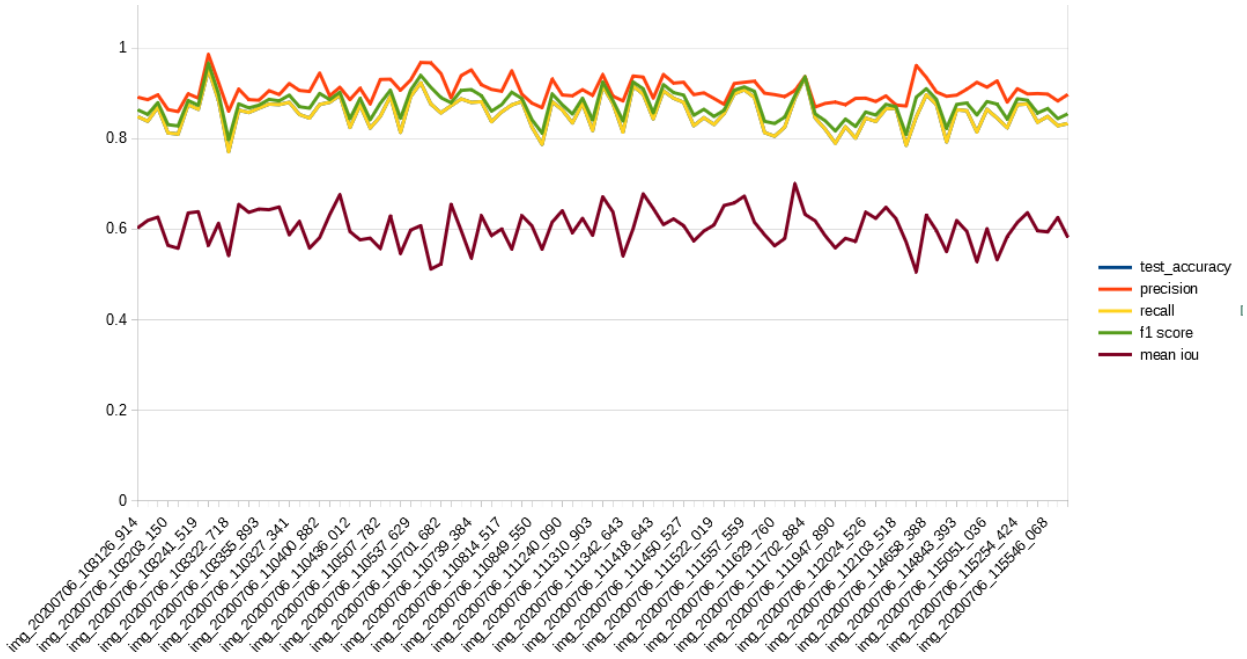


Figure 4.9 DeepLabV3 350 epochs results on the test set.

FRRN performance on the test set	
Training Set	592
Validation Set	159
Test Set	93
No. of epochs	150
No. of val. images/epoch	20
Accuracy	0.861114365591397
Precision	0.916946763440861
Recall	0.861114365591397
F1 score	0.879591666666667
IoU	0.607441677419355

Table 4.8 FRRN Experiment 1.

FRRN performance on the test set	
Training Set	592
Validation Set	159
Test Set	93
No. of epochs	300
No. of val. images/epoch	40
Accuracy	0.866323129032258
Precision	0.910106559139785
Recall	0.866323129032258
F1 score	0.880692569892473
IoU	0.625619107526882

Table 4.9 FRRN Experiment 2.

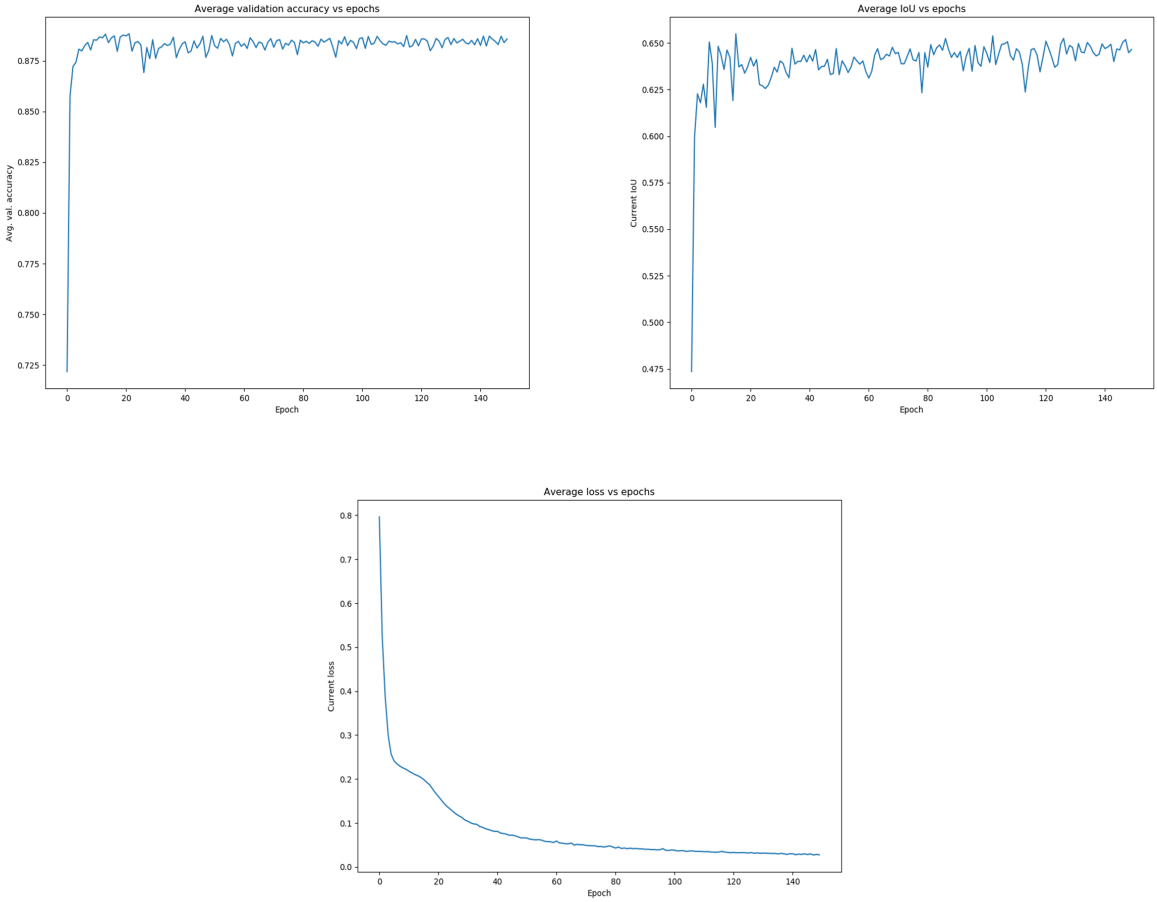


Figure 4.10 FRRN 150 epochs training monitoring. Top Left: Avarage validation accuracy vs epochs, Top Right: Average IoU vs epochs, Bottom: Average loss vs epochs.

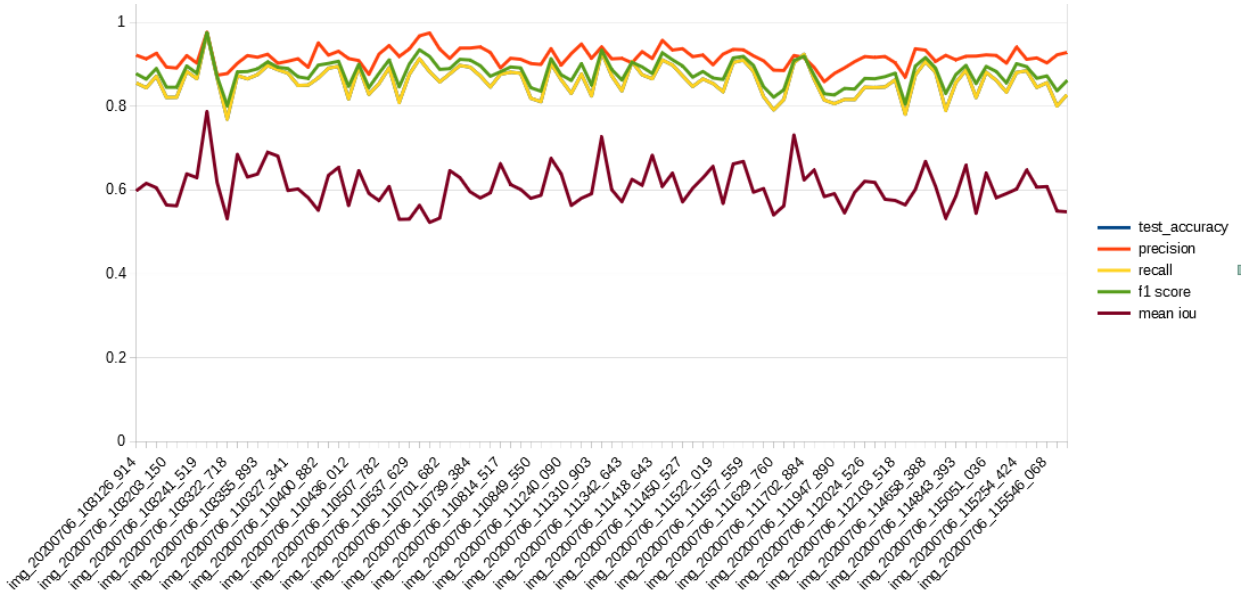


Figure 4.11 FRRN 150 epochs results on the test set.

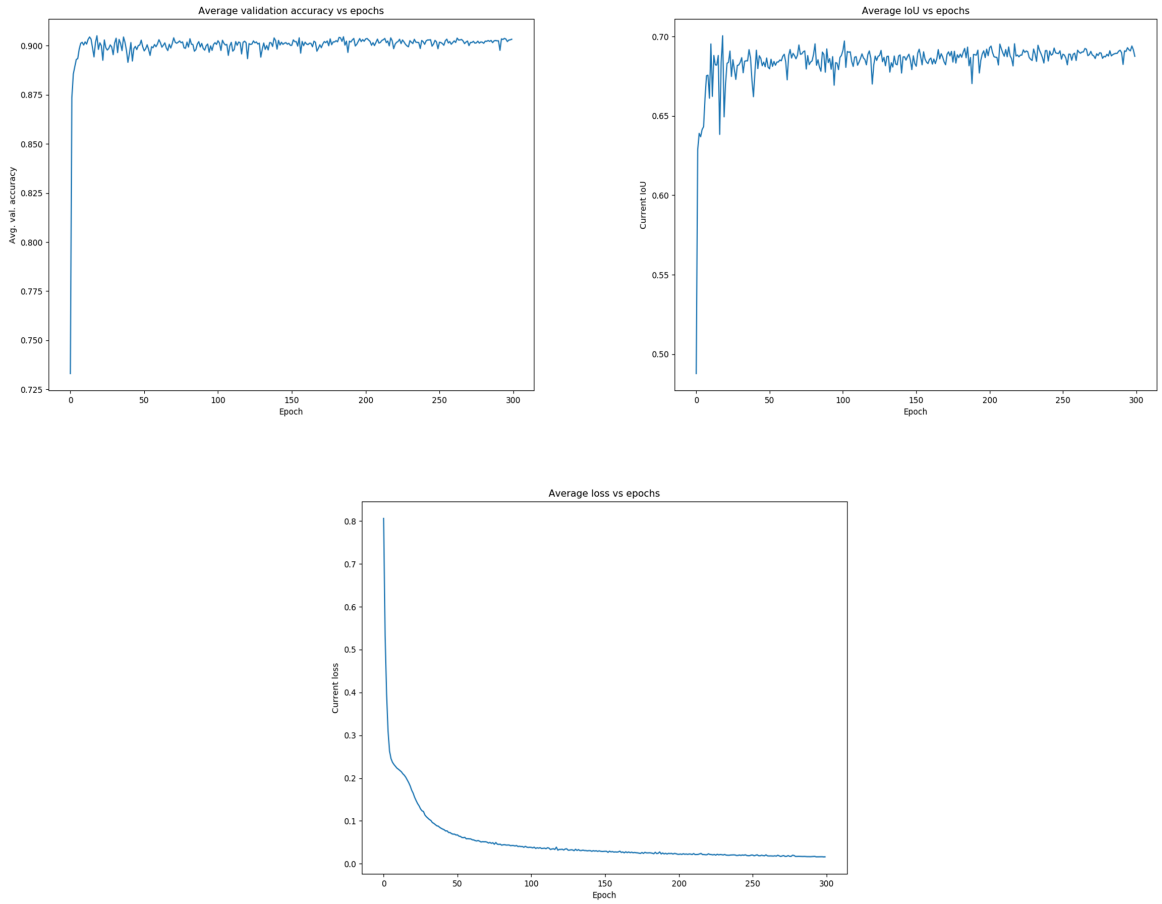


Figure 4.12 FRRN 300 epochs training monitoring. Top Left: Avarage validation accuracy vs epochs, Top Right: Average IoU vs epochs, Bottom: Average loss vs epochs.

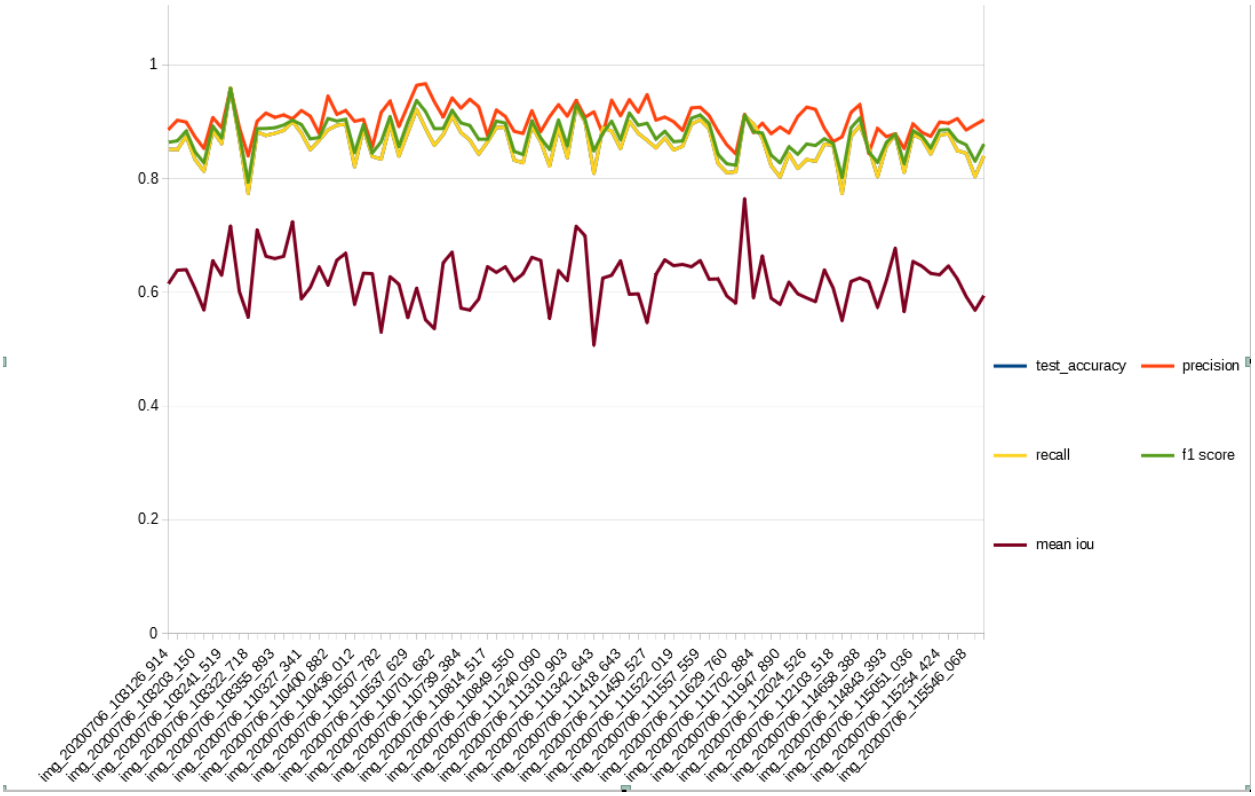


Figure 4.13 FRRN 300 epochs results on the test set.

Chapter Five

CIWA Web App

5.1 Architecture

As discussed in section 2.3, we designed the web app using microservices architecture. For the basic app functionality, we mobilized 4 containers, running React, nginx, Flask and mongoDB respectively. The containers are orchestrated by Docker. A detailed flowchart of our ecosystem can be found below [Figure 5.1].

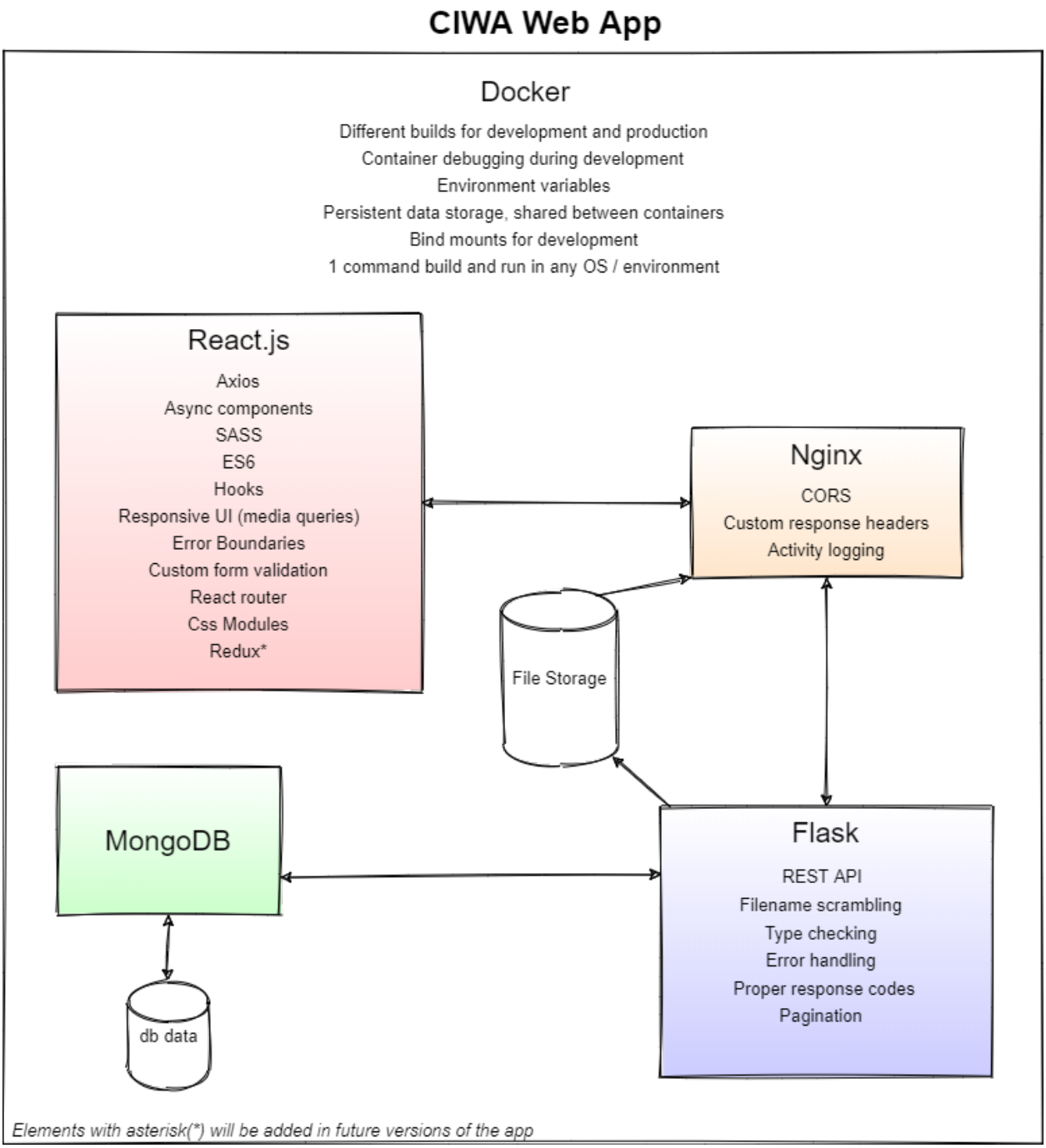


Figure 5.1 CIWA architecture

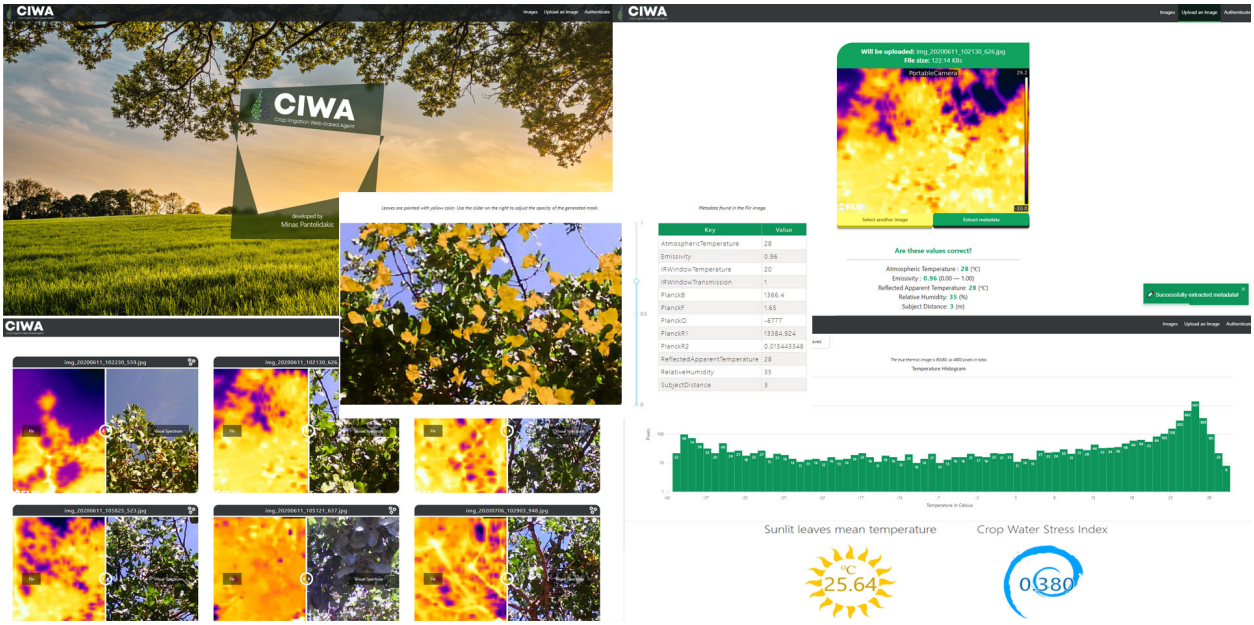


Figure 5.2 Components of the CIWA web app. Top Left: Home page, Top Right: Image upload screen, Bottom Left: Uploaded Image Gallery, Bottom Right: Temperature histogram, canopy temperature and crop water stress index, Middle: Metadata and FRRN generated mask of sunlit leaves.

5.2 Functionality

A user can upload a thermal image (currently supporting only FLIR AX8), have its metadata extracted and edited. After processing the metadata, the image is sent to the server and has its visible spectrum counterpart and raw thermal image reconstructed, and the underlying temperatures calculated. The trained FRRN model finds the sunlit leaves of the visible spectrum image, the prediction mask is downscaled and overlaid on top of the thermal image, and the mean sunlit leaf temperature and the Crop Water Stress Index are calculated. The app offers direct comparison between the thermal and visible spectrum images, temperature histograms, and the option to download temperature data in csv format. Some of the web app's components can be seen in Figure 5.2.

Chapter Six

Conclusion & Future Work

In this work, we investigated a method which utilizes convolutional neural networks and thermal imagery to identify sunlit leaves within images of trees, captured with a thermal camera.

The CNN-based methods we tested in this thesis preformed better than published literature methods, namely Histogram Gradient Thresholding (HGT) and Gaussian Mixture Models (GMMs). As shown in our experiments, HGT identifies most of the tree canopy, without highlighting sunlit leaves, while GMMs can easily missclassify artifacts of color profile similar to that of sunlit leaves. FRRN performed better than DeepLabV3, but also took a longer time to train.

6.1 Limitations of our approach

As discussed in subsection 3.3.1, the FLIR AX8 camera produces Visible spectrum and IR images that differ in resolution. Therefore, after sunlit leaves have been identified, the prediction has to be downscaled in order to be overlaid on top of the temperature array. This, inevitably, decreases the localization precision of our method.

We noticed that in about 1-3% of the captured thermal images, the FLIR AX8 camera produces and attaches irrelevant visible spectrum images, as shown in Figure 6.1. The exact

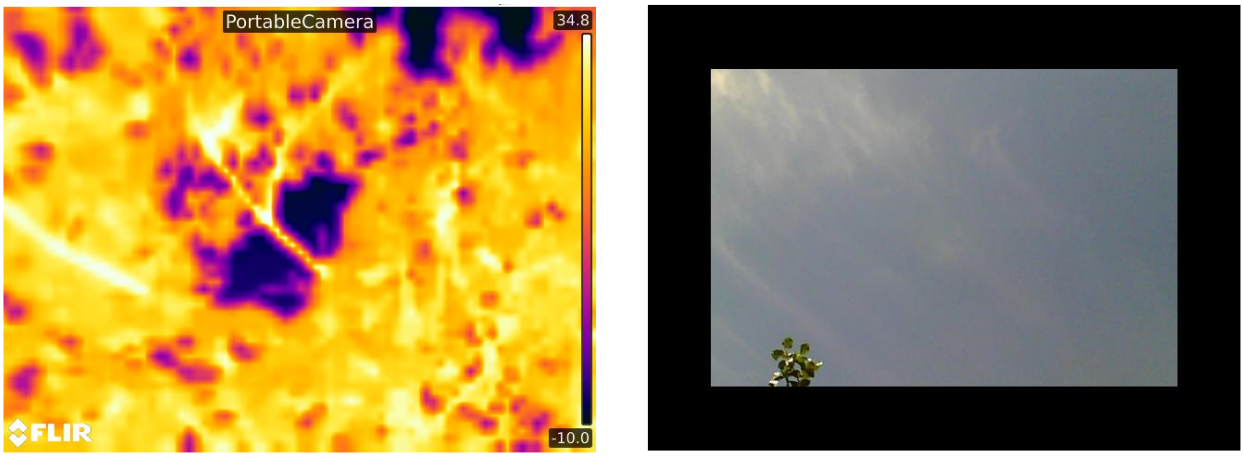


Figure 6.1 Flir AX8 bug example. Left: Thermal Image, Right: Visible spectrum image encoded into the thermal image metadata.

conditions under which the bug occurs are still unclear. We need to further investigate to be able to replicate this abnormal behavior. Nevertheless, these images have to be timely identified and excluded, since they yield wrong results. In an even more automated pipeline scenario, where the user does not get the chance to inspect the encoded visible spectrum image, this bug could pose a more serious problem.

We decided that performing clustering on the images before labeling them would significantly reduce the amount of time needed to generate a dataset of adequate size. Even though this proved to be true, the annotation accuracy is not pixel perfect.

Finally, we estimate that an even bigger dataset would further improve the performance of the CNNs.

6.2 Future work

We aim to deploy the implemented app to production via one of the cloud service providers that support containerized app architectures. We also plan on changing the way that our model is served from the backend, using Tensorflow serving.

A fast and straight-forward way to further enhance our dataset would be by utilizing the

CIWA Web App. Specifically we plan to give the user the option to fine-tune the generated mask and use the calibrated one as the new ground truth for the given image.

The current version of the developed application only supports the FLIR AX8 camera. We opt to implement multiple camera support in the future.

Flir also produces cameras for smartphones. Although these cameras do not yield the same level of temperature accuracy, they achieve promising performance. We are considering developing an app for smartphones using React Native/ Ionic Framework.

Finally we plan on testing our method on other crops, such as almonds and tomatoes. We are currently investigating the potential issues and challenges that these new crops would introduce.

The results obtained from this research project will also be presented to the California Pistachio Research Board and will be shared with University of California Farm Advisors, and other extension specialists.

REFERENCES

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Susstrunk. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34 (May 2012). DOI: 10.1109/TPAMI.2012.120.

[2] R. G. Allen, L. S. Pereira, D. Raes, and M. Smith. "Crop evapotranspiration: Guidelines for computing crop water requirements." In: FAO-56 (1998).

[3] *AQUASTAT - FAO's Global Information System on Water and Agriculture. Food and Agricultural Organization of the United Nations.* URL: http://www.fao.org/aquastat/en/geospatial-information/global-maps-irrigated-areas/irrigation-by-country/country/GRC.

[4] J. Bellvert, P. J. Zarco-Tejada, J. Girona, and Fereres E. "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 Agric* 15 (2014), pp. 361–376.

[5] Christopher M. Bishop. "Pattern Recognition and Machine Learning". In: (2006).

[6] C.M. Bishop. "Neural networks for pattern recognition." In: *Oxford university press* (1995).

[7] H. Caesar, J. Uijlings, and V. Ferrari. "COCO-Stuff: Thing and stuff classes in context." In: *arXiv:1612.03716* (2017).

[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation.* 2017. arXiv: 1706.05587 [cs.CV].

[9] P. D. Colaizzi, S. R. Evett, S. A. O'Shaughnessy, and T. A. Howell. "Using Plant Canopy Temperature to Improve Irrigation Crop Management." In: *Proceedings of the 24th Annual Central Plains Irrigation Conference, Colby, Kansas, February 21-22* (2012).

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Beneson, U. Franke, S. Roth, and B. Schiele. "The cityscapes dataset for semantic urban scene understanding." In: *CVPR* (2016).

[11] Kendall C. DeJonge, Saleh Taghvaeian, Thomas J. Trout, and Lousie H. Thomas. *Comparison of canopy temperature-based water stress indices for maize*. 2015. URL: http://digitalcommons.unl.edu/usdaarsfacpub/1500.

[12] Richard O. Duda and Peter E. Hart. "Pattern Classification and Scene Analysis". In: (1973).

[13] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserma. "The pascal visual object classes challenge a retrospective." In: *IJCV* (2014).

[14] Pedro Felzenszwalb and Daniel Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision* 59 (Sept. 2004), pp. 167–181. DOI: 10.1023/B:VISI.0000022288.19776.77.

[15] A. Giusti, D. Ciresan, J. Masci, L Gambardella, and J. Schmidhuber. "Fast image scanning with deep max-pooling convolutional neural networks." In: *ICIP* (2013).

[16] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: JMLR Workshop and Conference Proceedings, Nov. 2011, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a.html.

[17] V. Gonzalez-Dugo, D. Goldhamer, P. J. Zarco-Tejada, and E. Fereres. "Improving the precision of irrigation in a pistachio farm using an unmanned airborne thermal system". In: *Irrig Sci* 33 (2015), pp. 43–52.

[18] V. Gonzalez-Dugoa, P. J. Zarco-Tejadaa, and E. Fereres. "Applicability and limitations of using the crop water stress index as an indicator of water deficits in citrus orchards." In: *Agricultural and Forest Meteorology* (2014), pp. 94–104.

[19] C.C. Hau. "Handbook of pattern recognition and computer vision." In: *World Scientific* (2015).

[20] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learningfor image recognition." In: (2015).

[21] M. Holschneider, R. Kronland-Martinet, J. Morlet, and Tchamitchian P. "A real-time algorithm for signal analysis with the help of the wavelet transform." In: *Wavelets: TimeFrequency Methods and Phase Space* (1989), pp. 289–297.

[22] S. B. Idso. "Non-water-stressed baselines: a key to measuring and interpreting plant water stress." In: *Agricultural Meteorology* 27 (1982), pp. 59–70.

[23] S. B. Idso, R. D. Jackson, P. J. Pinter, R. J. Reginato, and J. L. Hatfield. "Normalizing the Stress-Degree-Day Parameter for Environmental Variability". In: *Agricultural Meteorology* 24 (1981), pp. 45–55.

[24] *Irrigation and Water Management Survey (formerly Farm & Ranch Irrigation). United States Department of Agriculture National Agricultural Statistics Service.* URL: https://www.nass.usda.gov/Surveys/Guide_to_NASS_Surveys/Farm_and_Ranch_Irrigation/.

[25] R. D. Jackson. "Canopy temperature and crop water stress." In: *Advances in Irrigation Research* 1 (1982), pp. 43–85.

[26] R. D. Jackson, S. B. Idso, R. J. Reginato, and P. J. Pinter. "Canopy Temperature as a Crop Water Stress Indicator". In: *Water Resources Research* 17.4 (Aug. 1981), pp. 1133–1138.

[27] M. E. Jensen. "Scheduling irrigation with computers". In: *Journal of Soil Water Conservation* 24(5) (1969), pp. 193–195. URL: https://eprints.nwisrl.ars.usda.gov/id/eprint/85.

[28] M. E Jensen and R. G. Allen. "Evaporation, evapotranspiration, and irrigation water requirements." In: Manual 70 (2016).

[29] M. E. Jensen, D. C. N. Robb, and C. E. Franzoy. "Scheduling irrigations using climate-crop-soil data." In: *Journal of the Irrigation and Drainage Division* 96(1) (1970), pp. 25–38. URL: https://eprints.nwisrl.ars.usda.gov/id/eprint/1207.

[30] Jongmin Jeong, Tae Yoon, and Jin Park. "Towards a Meaningful 3D Map Using a 3D Lidar and a Camera". In: *Sensors* 18 (Aug. 2018), p. 2571. DOI: 10.3390/s18082571.

[31] H. G. Jones, R. Serraj, B. R. Loveys, L. Xiong, A. Wheaton, and A. H. Price. "Thermal infrared imaging of crop canopies for the remote diagnosis and quantification of plant responses to water stress in the field." In: *Functional Plant Biology* 36 (2009), pp. 978–989.

[32] H.G. Jones. "Use of infrared thermometry for estimation of stomatal conductance as a possible aid to irrigation scheduling." In: *Agricultural and Forest Meteorology* 95 (1999), pp. 139–149.

[33] H.G. Jones, M. Stoll, T. Santos, C. de Sousa, M.M. Chaves, and O.M. Grant. "Use of infrared thermography for monitoring stomatal closure in the field: application to grapevine." In: *Journal of Experimental Botany* 53 (2002), pp. 2249–2260.

[34] I. Leinonen and H.G. Jones. "Combining thermal and visible imagery for estimating canopy temperature and identifying plant stress." In: *Journal of experimental botany* 55(401) (2004), pp. 1423–1431.

[35] K. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis. "Machine learning in agriculture: A review." In: *Sensors* 18(8) (2018), p. 2674.

[36] J.L. Monteith and G. Szeicz. "Radiative temperature in the heat balance of natural surfaces." In: *Quarterly journal Royal Meteorological Society* 88 (1962), pp. 496–507.

[37] R. Mottaghi, X. Chen, X. Liu, N. -G. Cho, S. -W Lee, R. Urtasun Fidler, and A. Yuille. "The role of context for object detection and semantic segmentation in the wild." In: *CVPR* (2014).

[38] Kevin P. Murphy. "Machine Learning: A Probabilistic Perspective (2012)". In: (2012).

[39] S. A. O'Shaughnessy, S. R. Evett, and P. D. Colaizzi. "Infrared Thermometry as a Tool for Site-Specific Irrigation Scheduling." In: Proceedings of the 26th Annual Central Plains Irrigation Conference, Burlington, CO. (Feb. 2014).

[40] S. A. O'Shaughnessy, S. R. Evett, P. D. Colaizzi, and T. A. Howell. "Automatic Irrigation Scheduling of Grain Sorghum Using a CWSI and Time Threshold." In: 5th National Decennial Irrigation Conference Sponsored jointly by ASABE and the Irrigation Association Phoenix Convention Center Phoenix, Arizona. December 5-8, 2010. (2010).

[41] G. Papandreou, I. Kokkinos, and P.-A Savalle. "Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection." In: *CVPR* (2015).

[42] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. *Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes*. 2016. arXiv: 1611.08323 [cs.CV].

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge." In: *IJCV* (2015).

[44] M. Salem, A. F. Ibrahim, and H. A. Ali. "Automatic quick-shift method for color image segmentation". In: *2013 8th International Conference on Computer Engineering Systems (ICCES)*. 2013, pp. 245–251.

[45] A.S.A. Salgadoe, A.J. Robson, D.W. Lamb, and D. Schneider. "A non-reference temperature histogram method for determining Tc from ground-based thermal imagery of orchard tree canopies". In: *Remote Sens.* 11 (2019), p. 714.

[46] A. dos Santos Ferreira, D. M. Freitas, G. G. da Silva, H. Pistori, and M. T. Folhes. "Weed detection in soybean crops using ConvNets. Computers and Electronics in Agriculture". In: 143 (2017), pp. 314–324.

[47] P Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. "Overfeat: Integrated recognition, localization and detection using convolutional networks." In: (2013).

[48] T.L.I. Sugata and C.K. Yang. "Leaf App: Leaf recognition with deep convolutional neural networks." In: *IOP Conference Series: Materials Science and Engineering* 273(1) (Nov. 2017), p. 12004.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[50] S. Taghvaeian, J. L. Chávez, W. C. Bausch, K. C. DeJonge, and T. J. Trout. "Minimizing instrumentation requirement for estimating crop water stress index and transpiration of maize." In: *Irrig Sci* 32 (2014), pp. 53–65.

[51] S. Taghvaeian, J. L. Chávez, and N. C. Hansen. "Infrared Thermometry to Estimate Crop Water Stress Index and Water Use of Irrigated Maize in Northeastern Colorado." In: (2012), pp. 3619–3637.

[52] C.B. Tanner. "Plant Temperatures." In: *Agronomy Journal* 55 (1963), pp. 210–211.

[53] Xuezhi Wang, Weiping Yang, A. Wheaton, Nicola Cooley, and Bill Moran. "Automated canopy temperature estimation via infrared thermography: A first step towards automated plant water stress monitoring". In: *Computers and Electronics in Agriculture* 73 (July 2010), pp. 74–83. DOI: 10.1016/j.compag.2010.04.007.

[54] W. Yang, X. Wang, A. Wheaton, N. Cooley, and B. Moran. "Automatic optical and IR image fusion for plant water stress analysis." In: *Information Fusion* (2009), pp. 1053–1059.

[55] C. Zhang, P. Zhou, C. Li, and L. Liu. "A convolutional neural network for leaves recognition using data augmentation." In: *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)* (Oct. 2015), pp. 2143–2150.

[56] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. "Scene parsing through ade20k dataset." In: *CVPR* (2017).

[57] H. Zhou, C. Yan, and H. Huang. "Tree species identification based on convolutional neural networks." In: *Intelligent Human-Machine Systems and Cybernetics (IHMSC)* 2 (Aug. 2016), pp. 103–106.