Exploring Deep Neural Networks for Plant Image Classification

by

Maryam Bafandkar

A Thesis submitted to the Faculty of Graduate Studies of
The University of Winnipeg
in partial fulfillment of the requirements of the degree of

Master of Science

Department of Applied Computer Science
University of Winnipeg
Winnipeg, Manitoba, Canada

Exploring Deep Neural Networks for Plant Image Classification

by

Maryam Bafandkar

Supervisory Committee

Dr. Christopher Henry, Surpervisor

(Department of Applied Computer Science)

Dr. Sheela Ramanna, Member

(Department of Applied Computer Science)

Dr. Murray Alexander, External Member

(Department of Physics)

This thesis is dedication to my mother, Mahnaz Borhanihaghigh, and my husband, Alireza Jahandideh.

Thank you for your sacrifice, support, and understanding.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Maryam Bafandkar
April 2022

Acknowledgements

I sincerely thank my supervisor, Dr. Christopher Henry, for the provided encouragement, patience, and invaluable contribution through the duration of my Master's program. I would also like to thank Dr. Christopher Bidinosti for providing me with the excellent opportunity to contribute to the TerraByte research group as deep learning architect. I am also grateful to Dr. Simon Liao and Dr. Sheela Ramanna for all their encouragement and helpful pieces of advice.

I would like to express my appreciation to my supervisory committee, Dr. Christopher Henry, Dr. Sheela Ramanna and Dr. Murray Alexander, for constructive criticism and helpful advice. I wish to thank all the members of the TerraByte research group for sharing their valuable ideas and feedback, and all the valued members of the Applied Computer Science Department and the Faculty of Graduate Studies.

I also take this opportunity to express a deep sense of gratitude and thanks to my family and my husband, who supported me all the time during my Master's journey.

I would also like to thank Mitacs, George Weston Ltd, the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Faculty of Graduate Studies for their financial support of this work. I would like to acknowledge the assistance and help of every person I met during my journey at the University of Winnipeg and in the Mitacs program.

Abstract

Automatically distinguishing different types of plant images is a challenging problem relevant to both Botany and Computer Science disciplines. Plant identification at the species level is a computer vision task called fine-grained categorization, which focuses on differentiating between hard-to-distinguish object classes. This classification problem is complicated and challenging because of the lack of annotated data, inter-species similarity, the large-scale features in appearance, and a large number of plant species. A plant classification system capable of addressing the complexity of this computer vision problem has important implications for society at large, not only in public computer science education but also in numerous agricultural activities such as automatic detection of cash crops and non-crop plants (called weeds). Furthermore, successful automation of crop and weed identification will lead to the reduction of chemical compounds currently used to eliminate weeds [15]. Deep Convolutional Neural Networks (CNN) can be a solution to perform this computer vision task. In this thesis, seven different CNN models are deployed to classify 1 million images - from the TerraByte dataset - of eleven very similar plant species [13]. This robust approach divides the problem into two main steps: the first step, called the generalist, identifies similar plants and separates them into different groups that contain indistinguishable plant species. The second step, called specialist, is used to classify plants within the groups of indistinguishable plants, including five weed and seven crop species, with high accuracy. The generalist-specialist CNN network shows that the hierarchical network outperforms simple CNN models in terms of accuracy and classifying similar plant images. The contributions of this thesis are the explored different CNN models and improved performance of those models by designing and implementing the generalist-specialist CNN models for classifying similar plant images.

Keywords: Plant identification, deep convolutional neural networks, hierarchical CNN, digital agriculture, plant images.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Botanists and farmers identify plant species based on their considerable knowledge and make decisions based on visible features, like the colour and size of each species [19]. Identification and classification of plant species is a challenging task because many species are difficult or impossible to classify without specialized knowledge. For example, focusing only on morphology – such as leaf size and shape – can be misleading due to many leaves being similar in appearance across species [70]. For personal gardens or greenhouses, this is not an issue as it is a simple task to label each species based on where they were planted, or potted [24]. As a result, removing unwanted species (called weeds) is a straightforward task. In contrast, most agricultural farms plant the same seed species (called a crop) throughout an entire field. On this scale, it is not feasible to manually remove weeds as it is a time-consuming and expensive task that requires many personnel to care for individual plants. Moreover, as was mentioned above, plant identification is also a difficult problem. For example, barnyard grass, an annual grassy weed found throughout North America, is visually very similar to wheat, which is one of Canada's most essential profitable and cultivated crops. Due to these issues, farmers use chemical methods, like herbicides to remove weeds. Thus, an automated solution to identify and classify plant species within an agricultural setting is essential to improve the quality and yield in large-scale agricultural settings, reduce waste, and prevent environmental harm through the use of fewer herbicides. Similarly, according to a report by the Food and Agriculture Organization of the United Nations, plants play a crucial role in the global food market [16]. Therefore, automated solutions for identification would also play a major role in human health. Over the last few years, machine learning (ML) approaches have solved many computer vision problems, such as semantic segmentation [35], facial recognition [6], and image annotation [73]. These developments were thanks to the development of large, labelled data sets, high-performance computing technologies, and powerful new deep learning (DL) algorithms [61, 83]. These DL algorithms have been

driven primarily by convolutional neural networks (CNN), and have been applied to different tasks in agriculture. [60] is one example presenting DL techniques for image-based plant identification at a very large scale. Full fruit detection systems [84] are another practical DL application to build an accurate, fast and reliable fruit detection system. [10] developed a deep learning system to learn discriminative features from leaf images along with a classifier for species identification of plants. A new CNN based framework has been designed by this study [74] for plant classification of various genotypes. They exploit the power of deep CNNs for the automatic joint feature recognition of plant genotypes images. Consequently, ML offers a foundation to develop automated systems that can identify crops and weeds of many diverse species that will provide an alternative to using herbicides to kill unwanted plants, which are harmful to the environment and humans.

1.1 Problem Definition

Despite many efforts [60, 10, 74, 84] using DL and new CNN models, developing automated plant classification algorithms with reasonable performance is still considered a challenging and unsolved problem. One reason for this is that plants have a very similar shape and colour representation across species [63]. Some researchers have attempted to solve this problem via transfer learning methods [51] and new CNN architectures [60, 10, 74, 84].

This thesis explores different DL models to classify plant images, namely, transfer learning with the VGGNet model [89], designing and developing CNN models, and implementing generalist-specialist networks inspired by hierarchical tree-based CNN architectures [2]. The dataset (described in Section 4.1) contains 12 images of plant species, in different stages of their lives (age). As shown in Section 4.1.3, certain plants are very similar in term of shape, color and texture. To remove noisy images, the dataset is preprocessed by relabeling and filtering out images by age (see Section 4.1).

1.2 Proposed Approach

In order to solve the classification of images containing plant species that are very alike, we explored seven different DL models. Firstly, following previous research on the application of trasfer learning on plant classification [59], transfer learning using VGGNet was investigated. Then three simple CNN architectures were designed to determine if a model with reduced parameters could solve the problem. These three DL models were able to boost the result of the first model explored. Based on different studies (see Section 3.4) on classifying images containing very similar objects, we designed a hierarchical tree-based CNN model

(generalist-specialist network). Here, the main idea is to create groups of classes, where each group consists of very similar classes. The first component is a network (called a generalist) for classifying which group (of similar classes) an image belongs to, and the second component contains several networks, each one responsible for classifying images within each group of similar classes. The generalist-specialist networks could successfully classify unrecognizable plant images with high accuracies.

We first preprocessed the data. Relabeling is done to decrease the number of classes, which simplifies the problem. For example, different types of beans are all relabeled as a bean. Next, we filtered out very young and very old images from the dataset. Some images are taken from the early growth stages of the plants in which the plant is barely visible through the soil. Then we split the samples into train, validation, and test data. Train data is used to train the models. Validation data is used as a cross-reference to check the performance of the models while the training process is running. The model does not see the validation and train data while training. After the training process is done, performance of each mentioned CNN architecture is assessed and compared with test data to measure how well the model can generalize.

Additionally, we evaluate and compare our CNN models with one produced by the company Sightline Innovation, which worked with a portion of the TerraByte[1] dataset [12].

1.3 Contribution

The contributions of this thesis described below.

- Processed a large plant dataset containing 1,207,829 labelled images from the TerraByte dataset, captured and produced during the period from 2020-04-01 to 2020-12-13.
- Explored and evaluated seven different CNN models to classify highly similar images of plant species from the TerraByte dataset. It includes wheat, canola, oat, bean, soybean, field pea, yellow foxtail, dandelion, wild bucketwheat, barnyard grass, smartweed, and Canada thistle.
- Designed and implemented a generalist-specialist CNN model implemented in a hierarchical tree-CNN structure to classify plant species that are very difficult for non-specialist humans to classify.
- For plant classification achieved 99.61% accuracy by using generalist-specialist networks.

[1] EMILI https://emilicanada.com/ and Terrabyte https://acs.uwinnipeg.ca/terrabyte/ website.

- Evaluated and compared those seven CNN models with an externally developed model trained on a portion of the TerraByte dataset. The results show the generalist-specialist networks have a better performance of plant identification by 12% in terms of accuracy from 45.65% to 57.18%.

1.4 Organization

The remainder of this thesis is organized as follows.

- Chapter 2 introduces neural networks and the training process. Additionally, the structure of the CNN model used in this thesis is defined.
- Chapter 3 is a literature review of digital agriculture and machine learning applications that solved the traditional agriculture problems. Furthermore, it reviews the DL models we worked on to solve the computer vision problem, like transfer learning using VGG16, CNN architecture and hierarchical tree-based CNN models.
- Chapter 4 introduces the dataset used for this work and explains the data preprocessing steps that have been applied to the dataset. Experimental hardware and software setup, and implementation details are also discussed. Additionally, an in-depth discussion of the structure of the CNN architectures used in this thesis is presented.
- Chapter 5 presents the results and analysis of the experiments performed in this thesis.
- Chapter 6 summarizes the work and experiments that have been developed in the thesis and discusses possible directions for future experiments.

Chapter 2

Theory

Neural networks are machine learning models that come in various forms and are constructed to solve a wide variety of tasks, including image classification [40]. In this section, we first describe the structure of *artificial neural networks*. We then describe *convolutional neural networks*.

2.1 Artificial Neural Network

Artificial neural networks (ANN) are inspired by the structure of the human brain, which includes billions of connected neurons [23]. Each neuron acts like a cell which has many inputs, and, after processing, a specific output will be produced [23]. For the remainder of this thesis, we will refer to an ANN as a neural network (NN). The neuron is called a perceptron, as depicted in Fig. 2.1. NN contain a large number of connected perceptrons. Each NN network can have different structures with different numbers of perceptrons.

2.1.1 Perceptron

As we mentioned, a perceptron receives many input signals [81]. We can propose a vector $\vec{X}$ to denote the perceptron input, where $\vec{X} = (x_0, x_1, \ldots, x_n)$, and, for each input value x_i, there is a corresponding weight w_i from the vector $\vec{W} = (w_0, w_1, \ldots, w_n)$. A perceptron with input $\vec{X}$ and weights $\vec{W}$ is modeled as

$$y = \sum_{i=0}^{n} w_i \times x_i,$$

y is the output of each perceptron. To have more flexibility and accuracy, we need a bias b, which shifts the result from the origin or zero value. For example, if there is no bias in the perceptron and the input is $\vec{X} = (0, 0, \ldots, 0)$, the output will be 0 no matter the value of the

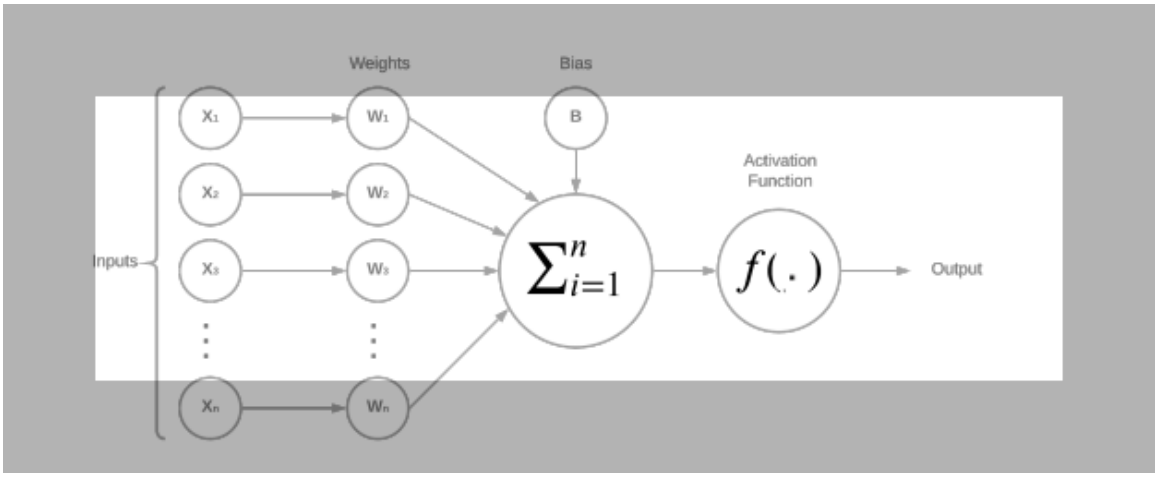

Figure 2.1 Structure of a perceptron

weights. A perceptron with input $\vec{X}$, weights $\vec{W}$ and bias b is formulated as

$$y = \sum_{i=0}^{n} w_i \times x_i + b.$$

2.1.2 Activation Function and Nonlinearity

Activation or nonlinear functions lead to increasing the approximation quality of the NNs. To make the result of the network nonlinear, the output of each perceptron should pass through an activation function, which determines whether to activate the neuron or not. There are three common nonlinear activation functions. The *sigmoid* function [61] (as defined in Eq. 2.1) limits the output between 0 and 1, *tanh* [61] (as defined in Eq. 2.2) limits the output between -1 and 1, and *rectified linear unit (ReLU)* [61] which in defined in Eq. 2.3. These functions are defined as:

$$\sigma(h) = \frac{1}{1+e^{-h}}, \tag{2.1}$$

$$\sigma(h) = \frac{e^{2h}-1}{e^{2h}+1}, \tag{2.2}$$

$$\sigma(h) = max(h,0). \tag{2.3}$$

ReLU is a very commonly used activation function. As shown in Fig. 2.2, ReLU is defined as

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x, \geq 0, \\ 0 & \text{otherwise .} \end{cases} \tag{2.4}$$

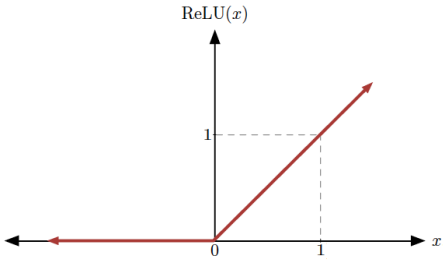

Figure 2.2 A plot of a ReLU non-linearity activation function

We will redefine the perceptron structure with an activation function to optimize the output of each perceptron as shown in Fig. 2.1 is shown.

2.1.3 Fully Connected Layers

Fully connected l layers is one NN architecture in which all nodes in the previous layer must be connected to all nodes in the following layer. We assumed l as the number of layers in a NN, $l^{(k)}$ as layer k and $m^{(k)}$ as the number of nodes in $l^{(k)}$. Let us considet two consecutive layers, $l^{(k-1)} \in \mathbb{R}^{m^{(k-1)} \times 1}$ and $l^{(k)} \in \mathbb{R}^{m^{(k)} \times 1}$. For these layers to be fully connected, the weight matrix connecting them would be defined as $w^{(k)} \in \mathbb{R}^{m^{(k-1)} \times m^{(k)}}$ [17]. The structure of a fully connected layer is shown in Fig. 2.3.

2.1.4 Interpretation of Gradient

A gradient is the collection of partial derivatives of a function. If we assume $\vec{X}$ as an input vector, a partial derivative of a function $f(\vec{X})$ is denoted as $\nabla f(\vec{X})$, where $\vec{X}$ is a vector of inputs [64]. For example, the partial derivatives of a simple multiplication function on two vectors $\vec{x}$ and $\vec{y}$ are defined as $f(x,y) = xy$ is as follows: $\frac{\partial f}{\partial x} = y, \quad \frac{\partial f}{\partial y} = x$. Partial derivatives represent the rate of change of a function with respect to the variables surrounding an immeasurably small region near a specific point [64], formulated as

$$\frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

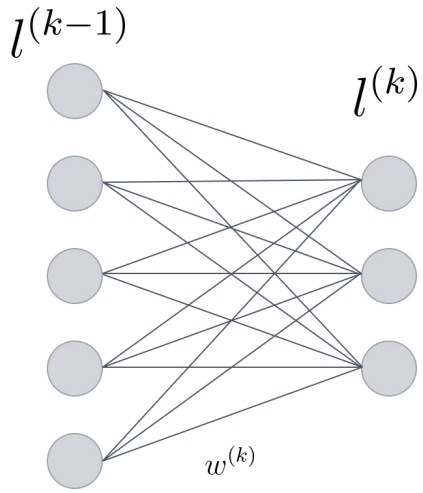

Figure 2.3 Graph representation of two fully connected layers, $l^{(k-1)}$ and $l^{(k)}$, connected by the weight matrix $w^{(k)}$

where operator $\frac{\partial}{\partial x}$ is applied to the function f and returns the derivative, and h is a number close to 0. Also, the vector of partial derivatives (gradient) ∇f is represented as $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] = [y, x]$. Partial derivatives are used in the process of learning in backpropagation and optimization algorithms, further described in the following sections.

2.1.5 Loss Function

A loss function is a method of evaluating the performance of a network. If the output of a NN, as a prediction, is wrong, the loss function will output a higher number, otherwise, it will output a lower number. One of the most common loss functions for image classification tasks is the cross-entropy (CE) function. The CE loss function [22] is calculated for each ground truth vector (y) and predicted vector ($\hat{y}$) where c is the total number of classes. It is computed as bellow where $y = (y_1, ..., y_c)$

$$\text{CrossEntropy Loss } = -\sum_{i=1}^{c} y_i \ln(\hat{y}_i)\,. \tag{2.5}$$

2.1.6 Back-propagation

In terms of addressing classification problems using supervised learning algorithms [79], like NN, we should work with a labelled dataset. Based on comparing the result of the loss function corresponding to an input image and its label, we can assess how close the network's output is to the correct label. The backpropagation algorithm looks for the minimum of the loss function with respect to all the weights in a NN using gradient descent [80]. Once the

output of the loss function is known, it will be used for the backpropagation algorithm using gradient descent to update weights. The gradient descent algorithm attempts to decrease the loss function value until the point where there is no change to the loss function value, also called convergence [77]. These steps are repeated again and again until the value of the loss function is less than some predefined threshold or epsilon value.

2.1.7 Learning Rate and Optimizer

A learning rate is the step size of each iteration in the gradient descent or other optimization algorithms. If the learning rate is too low, convergence will take a long time, but if the learning rate is too large, there might be no convergence at all. There are advanced and complex optimization algorithms for training neural networks, such as, momentum, root mean square propagation (RMSprop) [100], adaptive gradient algorithm (AdaGrad) [27] and adaptive moment estimation (Adam) [53].

In this thesis, we used Adam which is the most used optimization algorithm. Adam method stores an exponentially decaying average of past squared gradients v_j and keeps an exponentially decaying average of past gradients m_j, where j is the current iteration step. v_j and m_j are calculated as

$$m_j = \beta_1 m_{j-1} + (1-\beta_1)\,\delta_j,$$
$$v_j = \beta_2 v_{j-1} + (1-\beta_2)\,\delta_j^2,$$

where β_1 and β_2 are values close to 1 and δ_j is the partial derivative of the loss (L) with respect to the weights (w) at the iteration j. Also, the bias-corrected moment estimates are calculated as

$$\hat{m}_j = \frac{m_j}{1-\beta_1^j},$$
$$\hat{v}_j = \frac{v_j}{1-\beta_2^j}.$$

Then, the parameters are updated as

$$\theta_{j+1} = \theta_j - \eta \frac{\hat{m}_j}{\sqrt{\hat{v}_j} + \varepsilon}.$$

In this thesis, we have used the follwing values. $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-7}$, and learning rate $(\eta) = 0.001$.

2.2 Convolutional Neural Networks

Convolutional neural networks (CNN) are a variant NN architecture originally used in image processing applications. CNN are usually employed to solve computer vision-related tasks, like image classification [91]. It was also successfully deployed in natural language [39], and video processing applications [58]. The input to the network is an image represented as a 3-dimensional input layer with width, height and depth. The depth of the image is equivalent to the number of colour channels of the image. As Fig. 2.4 shows, the input layer has dimensions of $128 \times 128 \times 3$, corresponding to an RGB image of resolution 128×128.

CNN architectures can consist of one or more convolutional layers, optionally followed by downsampling, and then followed by one or more fully connected layers [30]. Fig. 2.4 shows an example of a CNN structure consisting of an input image followed by a pooling layer, then a convolution layer, another pooling layer, and a fully connected layer. CNN are typically structured in two parts. The first part, usually called feature extraction, employs combinations of convolutional and pooling layers. The second part, called classification uses fully connected layers [42].

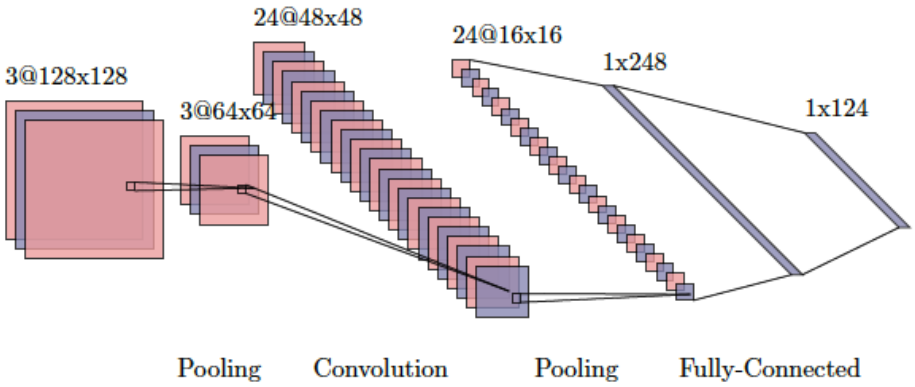

Figure 2.4 An example of a convolutional neural network structure for 128×128 image and 3 color channels with convolutional layer, pooling layers and fully connected layer [103]

2.2.1 Convolutional Layer

A convolution layer is the main part of a CNN that uses the convolution operation [29]. The output is the convolutional response of the given input with an associated spatial filter. Filters are learnable parameters in the format of matrices that respond to a specific pattern observed at some spatial position in the input. A convolutional layer uses filters to convolve input data to produce multiple feature maps, called feature extractions [42]. According to [28],

convolutional calculations are carried out based on the following formula:

$$\mathrm{y}\,(n_1,n_2,\ldots,n_m) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \ldots \sum_{k_m=-\infty}^{\infty} x\,(k_1,k_2,\ldots,k_m)\, f\,(n_1-k_1,n_2-k_2,\ldots,n_m-k_m)\,,$$

where n is the serial number [31] of output calculated by convolutional calculation; k is the serial number of input x; $(n-k)$ is the serial number of filter f : m is the dimension of the convolution.

In this study, we worked on $2D$ filters that move along two directions: the width and length directions. Therefore, the value of m equals two. The size of filters is a hyperparameter that considers different sizes for different CNN architectures [56]. Based on Fig. 2.5, the green 3x3 square on the blue square is the filter, and the blue square is the 4x4 region of the input image. A 3x3 filter size is the most common. There are several factors to choose the right size of filters, like the size of input image [18] and many learnable parameters [92]. The smaller the filters, the more parameters need to be calculated. Therefore, choosing a small filter size extends the training time and the number of learnable parameters, but instead, more features are extracted from input images. The depth of filters is called the activation map or the depth of the convolutional layer. The filters move on the input image from the top-left corner all the way to the bottom-right corner using the *stride* parameter. For a stride value of s, the filter moves s pixels over the input image. The input image passes through the network and gets convolved by more filters as it moves deeper, and the output becomes smaller and smaller. To preserve the original input size, *zero-padding* P is used. The number and size of filters, and the amount of zero padding (if any) for each convolutional layer, should be considered when designing a CNN architecture. If the zero padding is set to one, a one-pixel border is added to the image with a pixel value of zero. Fig. 2.5 depicts a convolution with a filter size of $3*3$ and zero-padding $P = 1$.

2.2.2 Pooling

The pooling layer can be used after a convolutional layer for the purpose of downsampling by reducing the spatial size of the features. It acts like a convolution calculation, similarly using filters and strides. But this time, instead of applying a weight, bias and activation map, a simpler function is used. An input image is divided into multiple non-overlapping rectangular sections, called *patches*, that are passed as input to the pooling function. This pooling function decreases the size of the output layer while preserving the most important information or features contained in the input layer.

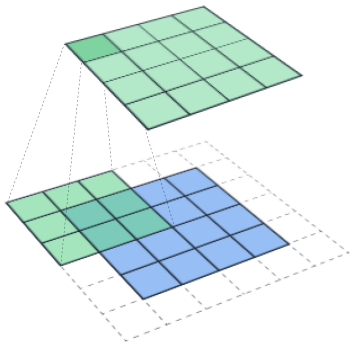

Figure 2.5 Convolution with a filter size of $3 * 3$ and zero-padding $P = 1$ over the input blue grid, where the green grid is the output of the convolutional layer [5]

We will define two important pooling functions to decrease the number of features with different methods or operations. The type of operation used for pooling layers defines its type.

1. Max pooling − This method uses the maximum operation and is commonly implemented for converting convolutional features of variable size images to a fix-sized embedding [21]. The maximum value in a channel within each patch is picked by max-pooling techniques. A positive effect of max pooling downsampling is that it extracts the learnable features with a maximum value.

2. Average pooling − This operation calculates the average value of cells on each patch per channel. It averages out the features in the neighbourhood, creating a blurring effect [75]. Max pooling completely wipes out the features while downsampling the feature map, and average pooling preserves the feature information, though it causes a blurring effect [75]. [90] shows that the average pooling method performs better on pictures on white background than max-pooling; on the other hand, max-pooling overperforms on dark background images compared with the average pooling method.

In this thesis, the max-pooling function has been used since it has a better performance on darker backgrounds [13] and can save computation cost [49].

2.2.3 Overfitting and Regularization

Overfitting is a common problem that affects NN. This is due to the fact that NN can learn too much from the images from the training dataset, a subset of the data. While NN are

training, they can memorize unique features and data-specific details found exclusively in the training dataset, mistaking these as general concepts shared across all similar data-input [36]. The result of this network on unfamiliar data, typically called a test dataset, does not perform as well as the training result. This is because the previously identified unique features memorized from training data do not exist in the new data. In other words, overfitting occurs where the weights of a NN converage too fast to the training dataset [41]. There are two main causes for the overfitting problem. One of them is insufficient data, which means that there are not enough variations of training data. In this case, a NN learnt from the limited data and extracted the data-specific details and features. Another one is having a complex model with many layers and parameters. As the complexity of a model increases (based on the number of parameters) the more specific features are extracted. When we have too much detail from the training data, the network memorizes exactly the input data.

2.2.4 Regularization

Although many parameters would make NNs to overfit, regularization is an effective way to obtain a model that generalizes well. To reduce overfitting without simplifying the network, we can use the regularization technique [33]. Fundamentally, this technique combats overfitting by penalizing the weight matrices of the network's nodes, thereby inhibiting the model's complexity during training by disrupting its ability to memorize [36]. If a regularization term is added, the model tries to minimize both the loss and complexity of the model. One common regularization method is to add a hyperparameter termed λ to the loss function, which influences how the weights are adjusted during the gradient descent algorithm. We look at the three types of regularizers, L1, L2 and dropout.

L1 Regularization

Overfitting occures when some weights are set to large values. One technique is to add a reguralization factor to the loss fucntion which penalizes large weight values. L1 regularization tends to push the regularized values towards zero. Therefore, it is effective in forcing the weights to become small or very close to zero. L1 regularization is defined as

$$L1 = \lambda \sum_{j=1}^{n} |w_j|$$

where $w \in \mathbb{R}^n$.

L2 Regularization

L2 regularization modifies the regularized values with a square term known as weight decay or ride regression. L2 regularization pushes the weight values towards zero (but not exactly zero). L2 regularization is defined as

$$L2 = \lambda \sum_{j=1}^{n} w_j^2.$$

Dropout

In addition to the L2 and L1 regularization, another famous and powerful regularization technique is dropout regularization. Dropout is an algorithm for training neural networks that was described at NIPS 2012 [43]. The training weights are deleted with probability $q = 1 - p = 0.5$, and the remaining weights are trained. The primary motivation behind the algorithm is to prevent overfitting problems by forcing neurons to be robust and rely on population behaviour rather than on the activity of other specific units [7].

2.2.5 Early Termination

NN are trained by presenting labelled input to the network and adjusting the network weights (using gradient descent) based on the loss function. An epoch denotes the situation where the entire training set has been passed through the network, and it is an important hyperparameter during the training process. Too many epochs can lead to overfitting of the training set, whereas too few may result in underfitting by the network [107]. Early termination is a method that allows you to specify an arbitrarily large number of training epochs and stop training once the model performance stops improving on a hold-out validation dataset.

Chapter 3

Literature Review

Since agriculture directly affects human health and the environment, many studies focus on using advanced technology and machine learning algorithms to solve current problems in agriculture. This chapter introduces *digital agriculture* and discusses how it can solve some traditional agricultural problems. Secondly, we introduce *machine learning* and *deep learning* concepts that have been applied to the agricultural field. Additionally, related studies in digital agriculture using machine learning algorithms are reviewed in this chapter.

3.1 Digital Agriculture

Weeds are the most costly category of agricultural pests. Worldwide, weeds cause more yield loss and increase farmers' production costs than insect pests. The word *weed* has been defined as an unwanted plant or a plant that is a pest in that it interferes with crop or livestock production [85]. In other words, any plant not intentionally grown requires management to prevent it from interfering with crop or livestock production called weeding. In this definition, a weed is a non-crop plant that can become a pest if not managed adequately [85]. A pest is a living organism, be it a plant, fungus, or animal, that is harmful to and threatens the life and existence of human beings and human concerns, livestock, crops, and forestry [32]. Bugs, fruit flies, leafrollers are some examples of pest categories.

To formulate a definition of the concept of digital agriculture, we consider the stages of agricultural development. Agriculture 1.0 is the first level of agriculture that was based on the use of manual labour (early 20th century). Agriculture 2.0 is called the "Green Revolution" (the late 1950s), when fertilizers, pesticides began to be actively used [55]. Fertilizer is a natural or artificial substance containing the chemical elements that improve the growth and productiveness of plants [66]. Moreover, a pesticide is any substance used to kill, repel, or control certain forms of plant or animal life that are considered to be pests. Pesticides

include herbicides for destroying weeds and other unwanted vegetation, insecticides for controlling a wide variety of insects, fungicides used to prevent the growth of moulds and mildew, disinfectants for preventing the spread of bacteria, and compounds used to control mice and rats [65].

Moreover, pesticide contamination moves away from the target plants, resulting in environmental pollution. Such chemical residues impact human health through environmental and food contamination [104]. It is generally accepted that pesticides play an important role in agricultural development because they can reduce the losses of farm products and improve the affordable yield and quality of food [3, 96]. Fertilizers and pesticides are a necessary evil for industrial agriculture [76]. They have adverse effects on soil, plants, environment, and human health [11]. Nevertheless, high efficiency in the agricultural sector is not currently possible without herbicides and pesticides [14]. Also, there is a high demand to produce more agricultural foods and products to meet the growing population. It is vital to have precise agriculture with less waste and sustainable outcomes [45].

After agriculture 2.0, using fertilizers and pesticides, came agriculture 3.0 called precision agriculture (the 1990s and 2000s), and agriculture 4.0 called digital agriculture (early 2010s) [55]. They utilize digital technologies and technical means in agricultural production making it possible to bring exact measurements to a new level when information on all agricultural processes and operations exist in digital form, and, at the same time, the transfer, processing and analysis of data are automated.

The development of technological advances has been growing year by year [88]. Using digital agriculture to classify plants is one of the trends [50]. Crop and weeds can largely vary within the same field. Identifying weeds and removing them from the field contributes significantly to the final yield. Digital agriculture allows scanning large areas of a plant and distinguishes between weeds and crops. In recent years, research into digital agriculture using machine learning methods has become an active area of study. [47] believed weeding is an effective way to increase crop yields. Their focus is on improving weed and crop recognition accuracy on weed dataset [59]. They found CNN [33] are favourable for multi-class crops and weeds recognition with limited labelled data in agricultural recognition tasks. Several studies in digital agriculture are focused on fruit detection from images taken from agricultural fields or orchards. [46] focuses on recognition and segmentation of overlapping apples. A model of a harvesting robot vision detector is studied in [106]. The model was improved to make it more suitable for the recognition and segmentation of overlapped apples. The method was tested by a random test set with 120 images, and the precision rate reached 97.31%, and the recall rate has reached 95.70%, which can meet the requirements of the apple harvesting robot's vision system [46]. [111] uses the same method using mask a model

called masked region CNN for strawberry detection. These works produce great accuracy ranging between 90% and 95% with good performance potential for generalization, showing that deep learning is a viable methodology for yield estimation.

We will explain more about machine learning, deep learning and specifically tree-CNN deep learning models and how they can play a significant role in developing digital agriculture.

3.2 Machine Learning

Two types of statistical machine learning (ML) techniques, such as supervised and unsupervised learning, have been utilized for precision agriculture [78]. Supervised learning entails learning a mapping between a set of input variables X and an output variable Y, called labels, and applying this mapping to predict the outputs for unseen data. Unsupervised learning is used on ML projects with the absence of class labels by clustering or partitioning of data [25]. Supervision invokes the idea of a teacher who guides the learning process. Typically this guidance comes in the form of labelled training examples that can be used to build a classification model. This external guidance is absent in unsupervised learning; thus, the process of building a model from the data can be done by clustering or organizing the data [25]. Some modern clustering techniques such as kernel k-means [93] and spectral clustering [105] are used for unsupervised learning.

In this study, supervised learning is used to classify crops and weeds. As is mentioned, for the supervised learning technique, a labelled dataset is required in order to transform the vast amount of data into helpful information [48]. One of the essential factors to improve the learning capability of computers is having a significant and clean dataset [94]. Creating an extensive dataset is the first step of each ML project. The process of data cleaning comes along with that, which is a process of removing incorrect, corrupted, incorrectly formatted, or incomplete data. After preparing the input data, we should work on ML algorithms and frameworks, specific tools to develop mathematical models that acquire knowledge without the explicit program but by seeing labelled examples.

According to [34], knowledge discovery using ML consists of six steps.

1. Problem specification. It involves gathering information about the application domain.

2. Problem understanding, which relies on previous knowledge about the domain that the experts have.

3. Data processing,which counts as an important step. It directly affects learning performance, as much as the success or failure of ML modelling. For example, augmentation

is a data processing technique producing more data with different effects, like zoom, horizontal, vertical effects. In this case, the network sees more images than the dataset has. Therefore, this data processing technique helps the network to learn from more diverse labelled images yielding better results, or prediction [99].

4. The machine learning process is the core methodology to obtain models for the extraction of patterns from input data. It involves the selection of specific algorithms and frameworks as well as the training and validation of the models.

5. Analysis of the discovered patterns and interpreting their meaning, called evaluation.

6. Exploitation related to implementing the trained machine learning model and tuning it to incorporate its knowledge in another application for further processing.

3.3 Deep Learning

The applications of Deep Learning (DL) and NN have remarkably increased due to a large number of advances in many research areas in the last two decades [62]. New papers in DL are published every month, where examples include image processing [68], computer vision [69], and object detection [37]. CNN's are the most common implementation of DL [61], being widely used in recent years in many scientific and industrial fields. CNN's have lately emerged as practical, effective tools in computer vision like object detection or image classification [52].

New technology and development in hardware, like improvements in graphics processing unit (GPU) technology and parallel computing help DL. CNN models, which previously needed months or weeks to train on a database with thousands of images, were able to be trained in just a few hours or days. In the past, training CNN models were restricted to using CPUs which made it nearly impossible to train a deeper model with large numbers of layers and training parameters in a realistic amount of time.

In 2012, the *AlexNet* model [57] won the ImageNet Large Scale Visual Recognition Challange(ILSVRC). It was able to outperform other algorithms in the ILSVRC image recognition and classification task. AlexNet introduced a deep learning model which consists of five convolutional layers and two fully connected layers. These two huge, fully connected layers produce model parameters of nearly 138 million parameters.

The success of AlexNet improved the speed and quality of DL research. In 2014, K.Simonyan and A. Zisserman from the University of Oxford presented the VGG architecture [89] which achieved 92.7% top-5 test accuracy in the ILSVRC. It won second place in the

classification and localization challenge in ILSVRC 2014. The VGG architecture is defined with 16 or 19 sequential convolutional layers. Since this DL model is a lot deeper than AlexNet, the size of filters used was decreased from 11×11 and 5×5 to 3×3 to reduce the number of learning parameters. The stride for each convolution layer is fixed to 1 pixel; the spatial padding of convolutional layer input is 1 pixel for 3×3 filter size. As Fig. 3.1 shows, five max-pooling layers carry out spatial pooling. Max-pooling performs over a 2×2 pixel window, with stride 2. The input to this DL network is a fixed-size 224×224 RGB image. The red number on the top-left of each box on Fig. 3.1 shows the input size for each convolutional block. All hidden layers are equipped with the ReLU [8] non-linearity function. In the end, it has two fully connected layers followed by a softmax for the output. The softmax function is a function that inputs a vector of n numbers and outputs a vector of n numbers which their sum equals to one. This DL model has 138 million parameters.

In digital agriculture, many applications take advantage of deep learning models to classify plants, crops, weeds, and plant disease detection [49]. Full fruit detection systems [84] are another practical DL application studied at Queensland University of Technology, Australia. The aim is to build an accurate, fast and reliable fruit detection system, which is a vital element of an autonomous agricultural robotic platform; it is a crucial element for fruit yield estimation, and automated harvesting [84]. Apple yield estimation was also possible from multi-geometry RGB images from an agricultural robot [9]. Disease detection applications employing CNNs have been recently researched in many research organizations, such as this study [71] in which the visual pattern identification of different diseases in leaf images was built by using deep learning. A similar approach was followed by [4], which used thousands of RGB images of banana leaves to identify the unhealthy banana yields. DL models have been built to detect the various weed species in the natural environment through the green colour and size of the leaves in crops and weeds [72]. The focus of this thesis is on plant classification using the VGG16 deep learning model.

3.4 Tree-CNN Deep Learning Models

Deep learning has significantly improved on visual recognition challenges recently. However, visual classification between different object categories can still be highly challenging. Some categories are more difficult to distinguish than others, based on their visual similarities. For instance, in the CIFAR100 dataset [108], it is easy to distinguish between an apple and a bus but harder to separate a lemon from an orange. Both lemons and oranges belong to a fruit category; they have a lot in common based on their appearances, while buses belong to another coarse category (vehicles 1) as defined within CIFAR100 [108]. We can say objects

in similar coarse categories are difficult to be separated. One solution to classify perceptually similar classes is developing dedicated classifiers [108]. For instance, a hierarchical model can be employed, where an initial classifier is used to group similar classes, and then a series of specialized classifiers can be used to make decisions within a group of very similar classes. Models of this type are called hierarchical, or tree-based structures. Fig. 3.2 shows, two similar plants that can barely be distinguished by human eyes.

In a hierarchical CNN classifier, the upper nodes classify the input images into subgroups [109]. Then, deeper levels classify deeper discrimination, for example, for fruit or human face classification problem. The upper nodes classify fruit or human face images into subgrouping like yellow-coloured objects together or human faces together [109]. Then, deeper nodes classify bigger differences, such as "lemon" v/s "orange" fruits or "old man," v/s "kid boy," human faces. Study [109] proves that hierarchical CNN models perform at par or even better than standard CNNs. In tree-based or hierarchical CNN structures, initial layers of a CNN from the top of structure learn very general features [87] that have been exploited for transfer learning [110, 86]. [108] used the hierarchical CNN's structure to deal with distinguishing the similar classes. They designed multi-layer hierarchical CNNs where an abstract higher-level network initially determines which subnetwork a sample should be directed to. Lower level networks are designed to find discriminating features amongst similar classes in the subnetwork. Each sub-network is called a class assignment classifier.

There are numerous examples using hierarchical classification to extract deeper features and detail of input images [101]. One of the earliest attempts of a CNN hierarchical approach [95] used transfer learning where objects are categorized hierarchically to improve network performance. In some studies, they used different classifiers for different levels of a tree CNN structure. This study [54] designed a hierarchial classifier by unifying decision trees and deep CNN layers. A deep CNN hierarchical structure on a large number of images was studied by [97, 44]. In some research, they used a hierarchical CNN-based classifier to build a two-stage classifier to separate easy and difficult classes [108].

In our work, we used a tree-CNN structure along with hierarchy labelling. It contains two levels. The single root is the first level of the tree structure that classifies different groups containing similar difficult objects. The second level contains a number of nodes that are equal to the number of difficult groups. In the next sections, we explain our tree-CNN design principle, network topology, and the algorithm used to implement the network.

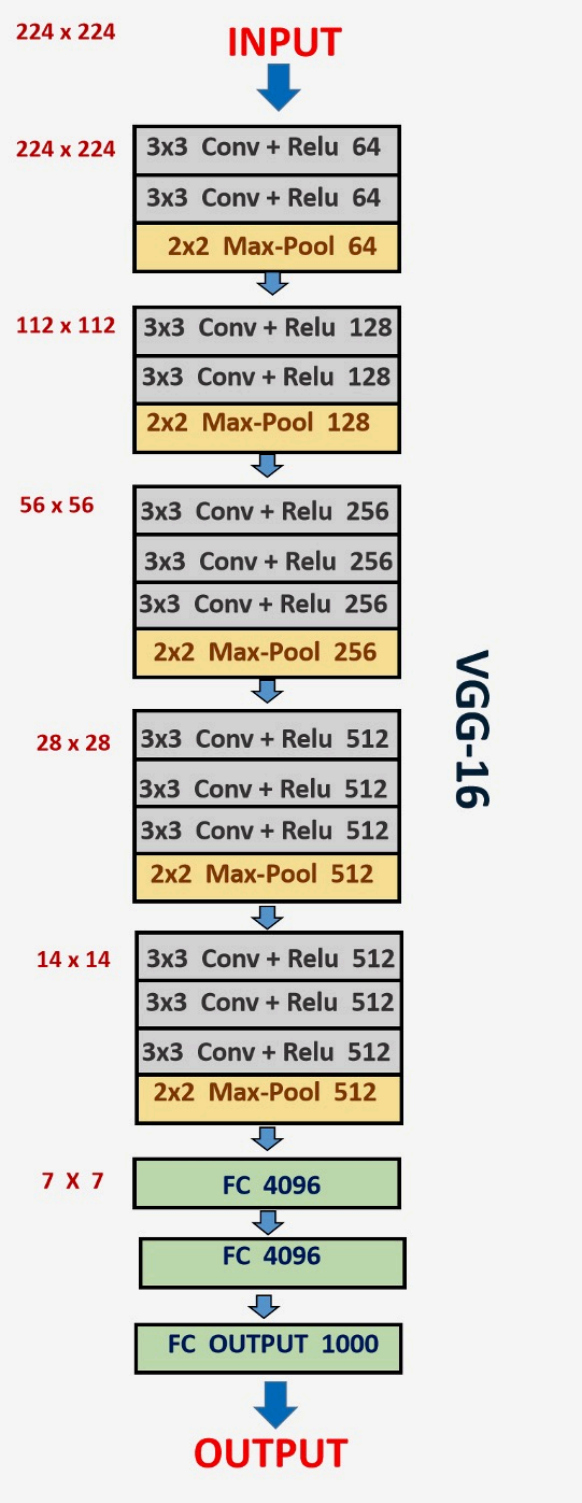

Figure 3.1 The convolutional layer parameters

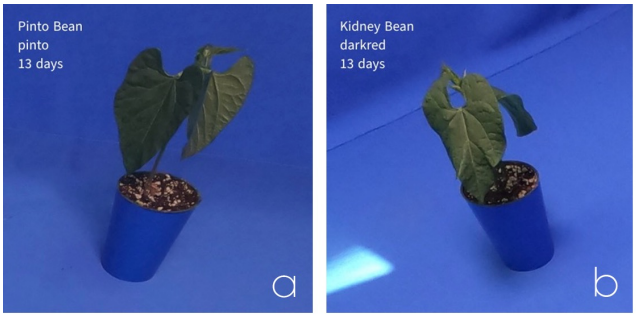

Figure 3.2 (a) Pinto Bean and (b) Kidney Bean

Chapter 4

Implementation Details

We discuss the implementation details in this chapter, divided into two parts: data preprocessing and models. Data preparation was one of the most important and time-consuming parts of this project. This chapter introduces the TerraByte[1] dataset [12], describes the process used to extract the samples used to produce results in this thesis, as well as the rationale behind these decisions. Furthermore, this chapter explains the DL models used in this study.

4.1 Dataset Preprocessing

Researchers from the TerraByte project have developed a robotic system that is capable of generating a large number of labelled images per day [12]. The results presented in this thesis were generated with data from the TerraByte project produced during the period April 1st - December 13th, 2020 from Lab-data dataset [13] . Table 4.1 shows the labels and the number of images per label. The following operations have been applied as a part of data preprocessing in this study:

- Re-labeling images were done to decrease the number of classes, hence, simpler classification problem;
- Filtering plants by age criteria to remove possible noise from the dataset;
- Grouping was done to generate a hierarchical structure of labels.

[1] Data can be downloaded from the EMILI and Terrabyte website.

Table 4.1 1,207,829 numbers of available labeled images from 2020-04-01 to 2020-12-13

Initial label	Count	Label
Pink Bean	23,447	Bean
Kidney Bean	47,731	Bean
Navy Bean	21,826	Bean
Black Bean	19,673	Bean
Pinto Bean	22,743	Bean
Cranberry Bean	22,892	Bean
FieldPea	55,553	FieldPea
Smartweed	104,660	Smartweed
CanadaThistle	112,658	CanadaThistle
Dandelion	108,713	Dandelion
Soybean	183,663	Soybean
Canola	229,517	Canola
Wheat	90,632	Wheat
YellowFoxtail	22,222	YellowFoxtail
Oat	27,025	Oat
BarnyardGrass	114,874	BarnyardGrass

4.1.1 Re-labeling Samples

To simplify the problem, we have merged different types of beans into one label. This is shown in Table 4.1 under *label* column. Fig. 4.1 shows one sample of each bean species to demonstrate their visual similarities.

Figure 4.1 Examples of different beans from the dataset, (a) Pinto Bean, (b) Kidney Bean, (c) Navy Bean, (d) Pink Bean, (e) Black Bean, and (f) Cranberry Bean

4.1.2 Age Filter

Images in the dataset are taken from seedlings to fully grown and flowering plants. Based on the visual inspection of samples, we have removed any image where the plant is a seedling or is too old. Different plants grow at different rates. Setting a fixed age to differentiate between seedlings and adult plants for all plants will not be a good solution. This is due to the fact that only an empty pot appears in an image if they are too young, and plants that are too old often have grown outside the boundary of the image. In the context of ML, it should be possible to train a model capable of classifying seedlings from adults or even flowering plants. In this thesis, we achieve this by sorting images by age and then visually looking for thresholds. Table 4.2 shows the minimum and maximum age for each plant; for example, for beans, we picked bean images from the age of 11 days old to 43 days old. Fig. 4.2 shows the growth process of Soybean as an example of how images have been sorted and inspected to find thresholds.

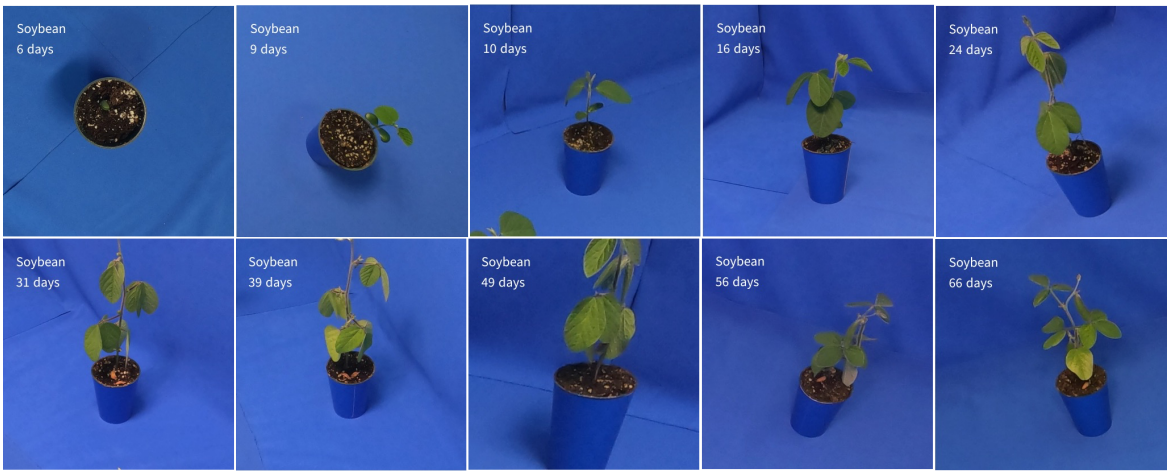

Figure 4.2 Examples of the growth stages of Soybean in different ages

Table 4.2 Applied age filter on the images and the number of resultant images

Label	Min. age (days)	Max. age (days)	Count
Bean	11	43	121,828
Field Pea	11	39	37,482
Wild Buckwheat	12	91	31,721
Smartweed	14	67	87,931
Canada Thistle	24	91	75,775
Dandelion	24	89	82,677
Soybean	14	66	130,054
Canola	14	65	149,303
Wheat	10	60	59,661
Yellow Foxtail	12	89	20,082
Oat	18	44	27,025
Barnyard Grass	18	43	61,591

4.1.3 Groupings

One goal of this thesis is to classify very similar images of plants. Thus, the dataset must be segmented into groups containing highly similar plants to train models to achieve this goal. Grouping the images was done based on visual similarities of plants. Technically, choosing the group is a hyperparameter. Different groupings can affect the results of the related experiments. Adjusting and finding the optimum grouping can be done either by simple iteration over the search space or using more advanced optimization methods. Doing so is out of this thesis's scope. The grouping used in this work is based on visual inspections done by the author. Figs. 4.3, 4.4, and 4.5 show the applied grouping, Table 4.3 shows the number of samples available for each group, and Table 4.4 is the final metrics on the extracted dataset to be used for the experiments.

Figure 4.3 Sample of plants in the G1 group, (a) Bean, (b) Field Pea, and (c) Wild Buckwheat

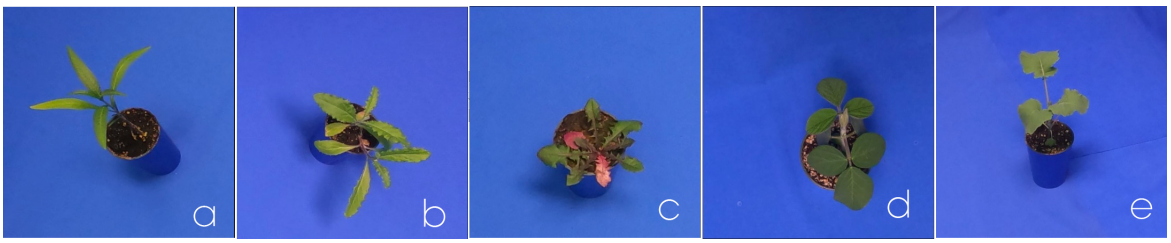

Figure 4.4 Sample of plants in the G2 group, (a) Smartweed, (b) Canada Thistle, (c) Dandelion, (d) Soybean, and (e) Canola

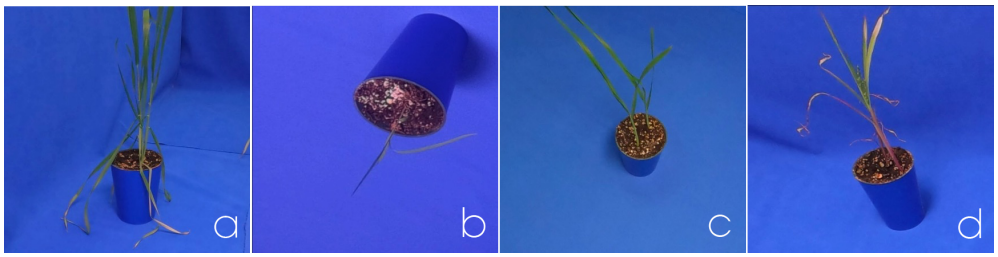

Figure 4.5 Sample of plants in the G3 group, (a) Wheat, (b) Yellow Foxtail, (c) Oat, (d) Barnyard Grass

Table 4.3 The number of samples in each group

Group	Count
G1	191,031
G2	525,740
G3	173,433

Table 4.4 Final dataset used for experiments with 883,050 plant images

Label	Group	Count
Wild Buckwheat	G1	31,721
Field Pea	G1	37,482
Bean	G1	121,828
Canola	G2	149,303
Canada Thistle	G2	75,775
Dandelion	G2	82,677
Soybean	G2	130,054
Smartweed	G2	87,931
Wheat	G3	59,661
Barnyard Grass	G3	61,591
Oat	G3	27,025
Yellow Foxtail	G3	20,082

4.2 Training and Test Data

From the total number of samples, we took 60% for training, 20% for the validation phase of the training, and 20% for testing the trained model. This data separation is kept constant for all experiments. As part of the process, we made sure that the 60-20-20 ratio was kept for all labels. This means, if, for example, 15% of all samples are labelled as Canola, then 15% of training samples, 15% of validation samples, and 15% of testing samples are labelled as Canola. The images are all randomized to remove the possible effect of ordering. The number of each label in the three datasets is shown in Table 4.5. Fig. 4.6 shows the distribution of different labels in the dataset. In total, 529,825 training samples, 176,610 validation samples, and 176,615 test samples are used in the experiments. Fig. 4.7 depicts the distribution of different groups divided by training, validation and test dataset.

Table 4.5 Training, validation, and test datasets

Label	Training Dataset	Validation Dataset	Test Dataset
Wild Buckwheat	18,818	6,273	6,273
Canada Thistle	45,162	15,054	15,055
Yellow Foxtail	11,835	3,945	3,946
Dandelion	49,396	16,465	16,466
Canola	89,581	29,861	29,861
Barnyard Grass	36,743	12,248	12,248
Bean	73,096	24,366	24,366
Soybean	78,032	26,011	26,011
Smartweed	52,662	17,554	17,554
Wheat	35,796	11,932	11,933
Field Pea	22,489	7,496	7,497
Oat	16,215	5,405	5,405
total	529,825	176,610	176,615

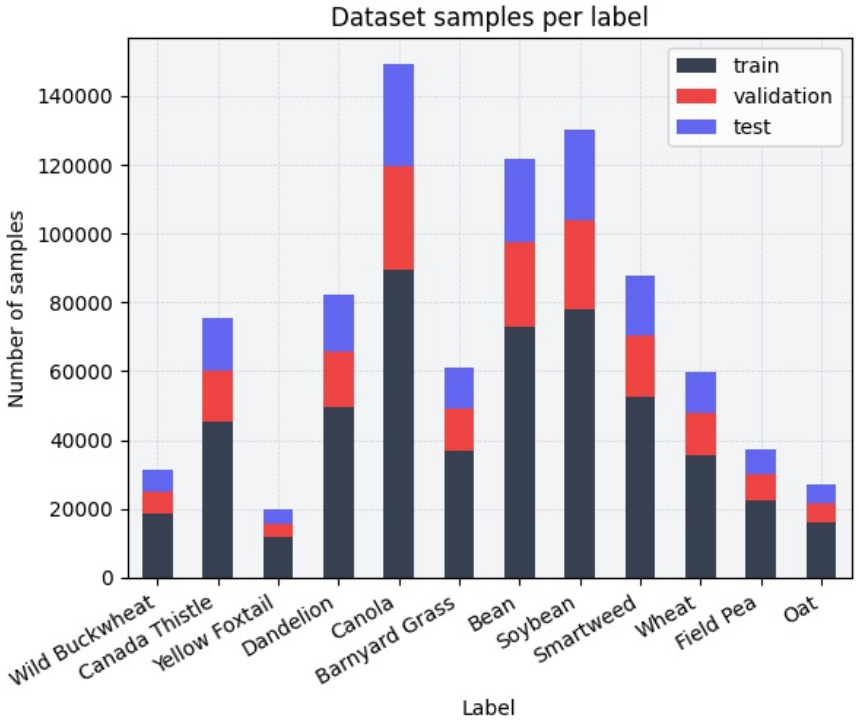

Figure 4.6 Distribution of different labels in data

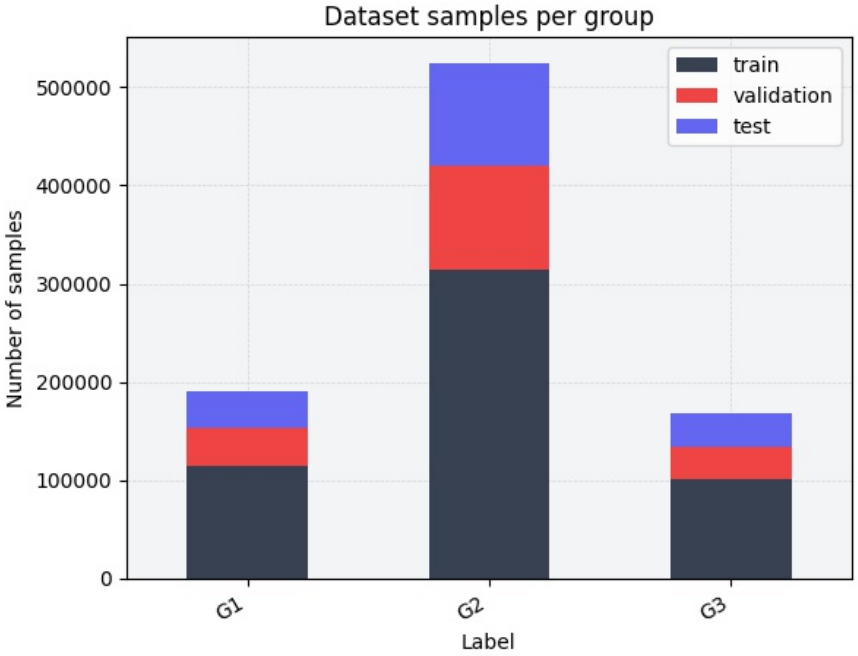

Figure 4.7 Distribution of different groups in the dataset

4.3 Experiment Hardware and Software Setup

The models were implemented using Python, Tensorflow [1] library. Experiments were performed using two systems. The first was equipped with 4x NVIDIA Tesla V100s, and 8x NVIDIA RTX A6000. The software and hardware used are listed in Table 4.6.

Table 4.6 Experiment setup

	Setup A	Setup B
Tensorflow	v 2.4.2	v 2.4.2
CUDA	v 11.3.1	v 11.3.1
GPUs	4 x NVIDIA Tesla V100	8 x NVIDIA RTX A6000
CPUs	24 x Intel Xeon	80 x Intel Xeon
RAM	88 GB	470 GB
DISK	200 GB NVMe	200 GB NVMe

4.4 Models

This section introduces the different deep learning models and architectures implemented in this thesis. The first model is *VGG-16 using transfer learning* which is a common architecture in image classification applications [98]. The second model considered in this work are two hand-crafted CNN architectures. Third, we designed and developed *a tree of CNN model*, called *generalist-specialist* architecture.

4.4.1 Transfer Learning

Following previous research on the application of transfer learning techniques on plant classification [51], we have used a VGG model which was trained on the ImageNet database [26]. This model was trained on a dataset of images from 1000 different categories. The idea is to keep the knowledge of the network (such as learned representations of color, edges, and texture) and transfer this knowledge to a new domain. As Fig. 3.1 shows, VGG-16 contains 5 convolutional blocks including 16 convolutional layers and 5 pooling layers. The network has been trained on the ImageNet dataset. To transfer the knowledge of VGG-16 model on the ImageNet dataset to our model, we removed the last 3 dense layers of the architecture and attached two dense layers at the end of the network as the new classifier. This is because

the original model has been trained on the ImageNet, which contains 1000 classes. In our case, the number of classes is less. To ensure better generalization and avoid overfitting, we added a dropout layer between two dense layers. As Fig. 4.8 shows, this model has been divided into two parts. The first part consists of frozen CNN layers which have been trained using the VGG-16 model, called *pretrained VGG-16 model*. The classifier part is the second part that we developed to address the plant classification problem on our plant dataset. Total parameters of the model is 14,983,500 from which 268,812 are trainable and 14,714,688 are frozen (pre-trained).

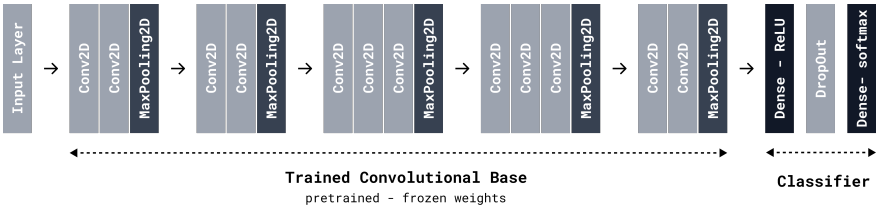

Figure 4.8 Transfer learning with pretrained VGG16 model

4.4.2 Simple CNN models

We designed the simple CNN models to solve the classification of similar plant images to see if a network with fewer parameters can have the same or even better performance when compared with the VGG-16 model.

The transfer learning uses pretrained model that has been trained on the ImageNet database containing more than a thousand objects [102] used for object classification problems, but our target specifically is plant recognition. For those reasons, we have included a simple model with three variations that are similar to VGG-16 in their architecture. These models are built by putting together a series of convolution blocks followed by multiple fully connected and dropout layers as well as a final activation layer. Fig. 4.9 shows two different simple CNN models used in this thesis. One model has three convolution blocks, and the other has five blocks. These models use random initial weights, which are then trained on our dataset from scratch instead of using unknown pre-trained models to train and optimize the simple CNN models.

4.4.3 Tree of CNNs

Plant images are visually similar, and there are many different types of plant species recognized in nature and the agriculture industry. Since transfer learning techniques usually work on many different objects (animals, plants, human faces, etc.), it might not be accurate

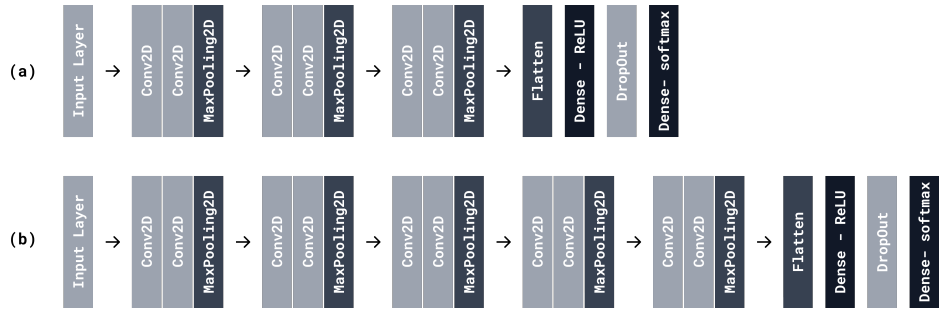

Figure 4.9 CNN models (a) with 3 convolution blocks, (b) with 5 convolution blocks

enough when we are dealing with a large number of specific similar plant species. We need to design CNN architectures to be trained specifically on plant images and recognize highly similar plant species. We also show in the Section 5 that the tree-CNN architecture can beat the transfer learning techniques to recognize plant classes in our dataset. As the goal of this study is to explore different approaches and compare them, we have considered developing a tree of CNNs. This approach has been tried before on different domains [82, 67]. The idea is to group the final classes into families. Classes in a group are more similar to each other than the classes in other groups. One CNN is trained to classify the group of the input image, and a number of CNNs, as needed, are trained to classify the classes in each group. For example, if we have six classes, we may put them into two groups based on their visual similarities. In this case, three CNNs are needed. One CNN to classify the input images into groups (G1 and G2). This is called the *root CNN*. Two CNNs classify input images into classes that are grouped as G1 and G2, respectively. These CNNs are called *leaf CNNs*. The CNNs form a tree-like structure as shown in Fig. 4.10. These models are also referred to as *generalist-specialist* since the root CNNs can be considered as a general classifier and leaf CNNs are the specialized classifiers.

In this thesis, we have developed three tree models, one using transfer learning of VGG-16, and two using the models discussed in Section 4.4.2. How we decide to group the classes is a hyperparameter. It has a direct effect on the performance of the models. We have not tried different groupings in this study and left it to future research. We visually inspected the images from the dataset and made three groups of plants based on their visual similarities. The groups are G1, G2, and G3, with compositions shown in Table 4.4. Training of these CNNs is performed individually. To train the generalist CNN, we have used the relabeled dataset with $G1$ to Gn labels, where n is the number of groups ($n = 3$). To train each specialist CNN, we have filtered the training and validation datasets to contain only the generalist images distinguished for the related group. Testing is performed on individual networks using the following logic:

Algorithm 1 Tree of CNNs Algorithm

INPUT: X Input image
$group = Generalist(X)$
if $group$ is $G1$ **then**
  $\hat{y} = SP1(group)$
else if $group$ is $G2$ **then**
  $\hat{y} = SP2(group)$
else
  $\hat{y} = SP3(group)$
end if
OUTPUT: $\hat{y}$ predicted label

In Chapter 5, we present the results of each model and show the performance using the confusion matrix and loss/validation diagrams.

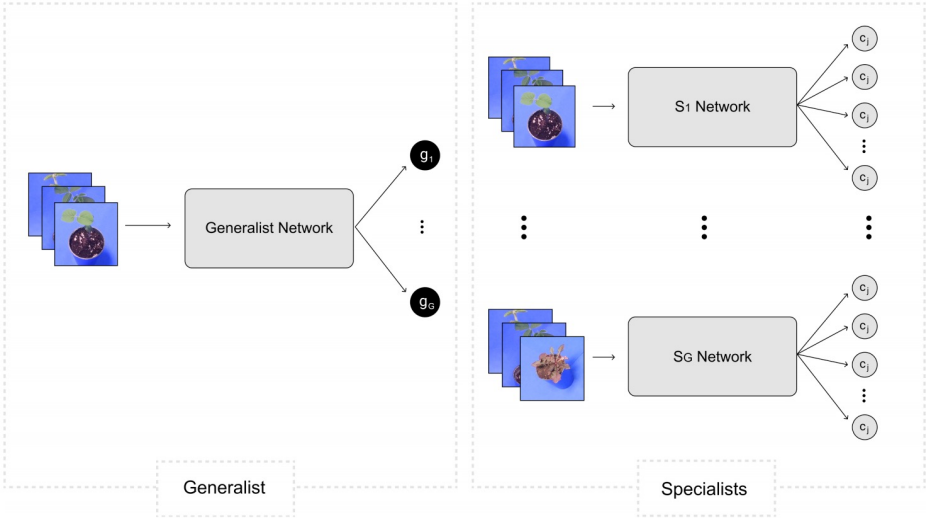

Figure 4.10 Structure of the generalist-specialist model

Chapter 5

Experiments and Results

This chapter introduces different experiments on the Terrabyte dataset [12]. Each experiment is focused on a different CNN model with parameters. The architecture of each model is explained, along with the results and subsequent analysis.

5.1 Evaluation Metrics

In this section, we describe the metrics used to evaluate the performance of the models, namely: precision, recall, F1-score, and accuracy. For a binary classification where the classifier predicts positive or negative, the *precision* is the fraction of correct positive predictions. It answers the question: *what proportion of predicted positives is truly positive?* It is defined as $\frac{TP}{TP+FP}$, FP is the number of incorrectly predicted positives (false positives). *Recall*, on the other hand, answers the question: *what proportion of actual positives is predicted positive?* It is defined as $\frac{TP}{TP+FN}$, FN is the number of incorrectly predicted negatives (false negatives). *Accuracy* answers the question: *what proportion of images — both positive and negative — were correctly classified?* It is defined as $\frac{TP+TN}{TP+FP+TN+FN}$. Both precision and recall are well known metrics, but F1-score is also important to compare two classifiers. It is a way to combine precision and recall into a single number. It is defined as $\frac{2\times precision\times recall}{precision+recall}$. F1-score is the harmonic mean of precision and recall that gives larger weight to lower numbers [20].

In multi-class classification, metrics such as precision, recall, and F1-score are defined for each class. For example, assuming that the classifier has three classes A, B, and C in an image classification problem, the precision for class B is the number of correctly predicted B photos out of all predicted B photos. The recall for class B is the number of correctly predicted B photos out of all actual B photos. The F1-score for class B is the harmonic mean of precision and recall for class B. For multi-class problems, the per-class metrics can be combined into single numbers. *Macro average* is the arithmetic mean of per-class metrics

where the values of precision, recall, and F1-score are added for all classes and divided by the number of classes. The macro average gives equal weights to each class. In *weighted average* each class is weighted by the number of samples of that class. Another way to calculate precision, recall, and F1-score for a multi-class problem is called the *micro average*. To calculate the micro average precision, we look at whole predictions. TP is the number of correct predictions through all classes, while each prediction error becomes a FP. The micro average recall is the same as the micro average precision. Since all the classes are together, FN is also the number of prediction errors. The micro average F1-score will also be equal to the micro average precision and recall because in the formula for F1-score the values factor out. The classifier's overall *accuracy* is defined as the proportion of correctly classified samples out of all samples, which is equal to the micro average F1-score [38]. For that reason, we have replaced the micro average F1-score with accuracy in tables like Table 5.1.

5.2 Experiment Results

In Chapter 4, the different deep learning models and architectures implemented in this thesis are discussed. They are based on three defined CNN models, including transfer learning, simple CNN models and the tree of CNN models. We designed seven different experiments to explore solving similar plant classification problems. This section will introduce the seven different experiments with relevant information such as the value of parameters or the number of convolution layers. Additionally, for each experiment, the test results and evaluation metrics will be presented in this section.

5.2.1 Experiment E001 (Transfer Learning with VGG16)

The model used in this experiment is VGG-16. We have used the VGG16 model trained on the ImageNet dataset. After importing the model, all layers were frozen, and then we fine-tuned the last layer by adding a flattening layer and a fully connected layer followed by a dropout layer. The last block is trained on the training dataset. The training batch size is 128, and the maximum epochs allowed is 45. The Adam optimizer is used for training with a learning rate of 0.001. The training and validation accuracy and loss of the model while training is included in Appendix A. The model is tested on the test dataset described in Section 4.2 and the confusion matrix is shown in Fig. 5.1. The overall accuracy of the predictions is 97.86% for 176,160 test samples, while the recall and F1-scores were close to each other for all classes. The highest precision is 99.12% for Bean and the lowest precision

is 94.74% for Yellow Foxtail as shown in Table 5.1. Note that support column is showing the number of samples in each class.

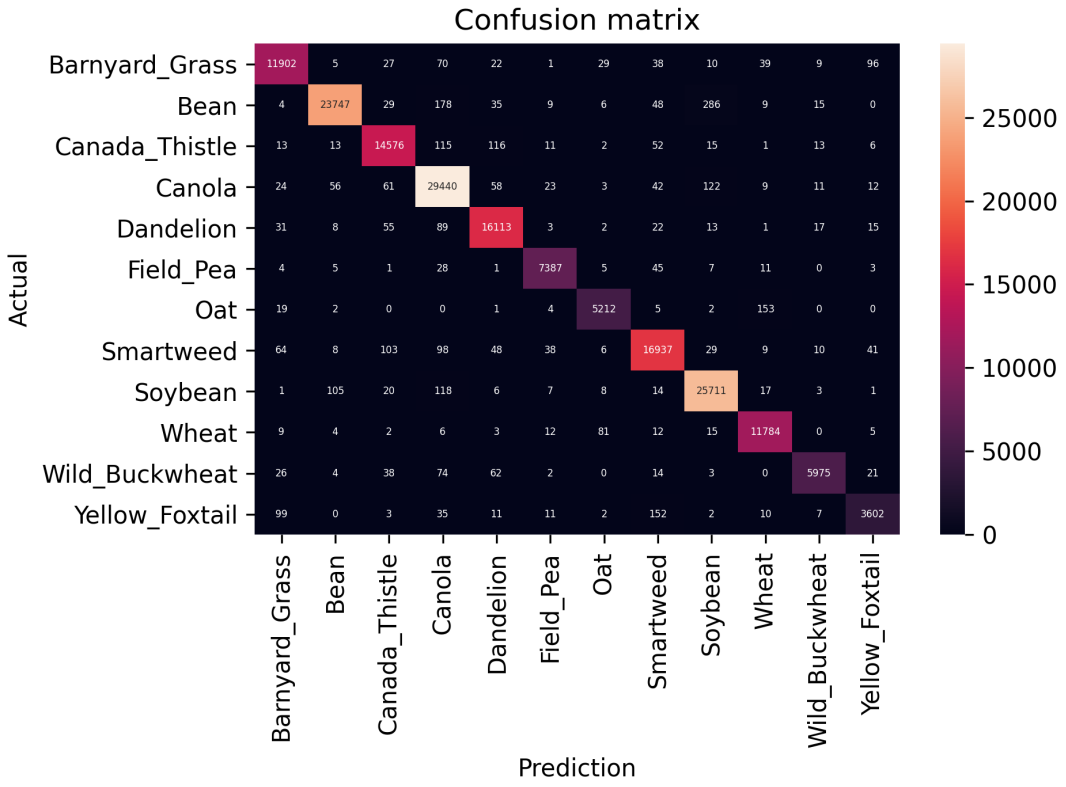

Figure 5.1 Confusion matrix of E001 model

Table 5.1 Evaluation metrics for E001 model on test dataset

	precision (%)	recall (%)	F1-score (%)	support
Barnyard_Grass	97.59	97.18	97.38	12248
Bean	99.12	97.46	98.28	24366
Canada_Thistle	97.73	97.61	97.67	14933
Canola	97.32	98.59	97.95	29861
Dandelion	97.80	98.44	98.12	16369
Field_Pea	98.39	98.53	98.46	7497
Oat	97.31	96.55	96.93	5398
Smartweed	97.45	97.39	97.42	17391
Soybean	98.08	98.85	98.46	26011
Wheat	97.85	98.75	98.30	11933
Wild_Buckwheat	98.60	96.08	97.32	6219
Yellow_Foxtail	94.74	91.56	93.12	3934
accuracy			97.86	176160
macro avg	97.66	97.25	97.45	176160
weighted avg	97.86	97.86	97.86	176160

5.2.2 Experiment E002 (5 convolution blocks and 3x3 filters)

In this experiment, along with E003 and E004, we studied three similar simple CNN models. We were inspired by the VGG-16 model that starts with simple CNN blocks. For E001 the model is formed by 5 convolution blocks described in Section 4.4.2. We want to explore the CNN with just 5 and 3 convolutional blocks to make it simpler and have fewer parameters to learn. Since training occurs with a large dataset, a model smaller than VGG-16 (in terms of number of weights) will train faster and more efficiently. The convolution operation is performed with strides of 3×3 for horizontal and vertical movements, respectively. Then a flattening layer is used, followed by 1 fully connected layer (1024) with ReLU activation, 1 dropout layer (0.1), and a final fully connected layer with a softmax activation. The training batch size is 128, and the maximum epochs allowed is 45. The Adam optimizer is used for training with a learning rate of 0.001. The training and validation accuracy and loss of the model while training is included in Appendix A. The model was tested on the test dataset, and the confusion matrix is shown in Fig. 5.2. The overall accuracy of this experiment is 99.63% for all classes. Table 5.2 shows the lowest precision goes for Yellow Foxtail and Oat with 98.36% and 98.74%, respectively.

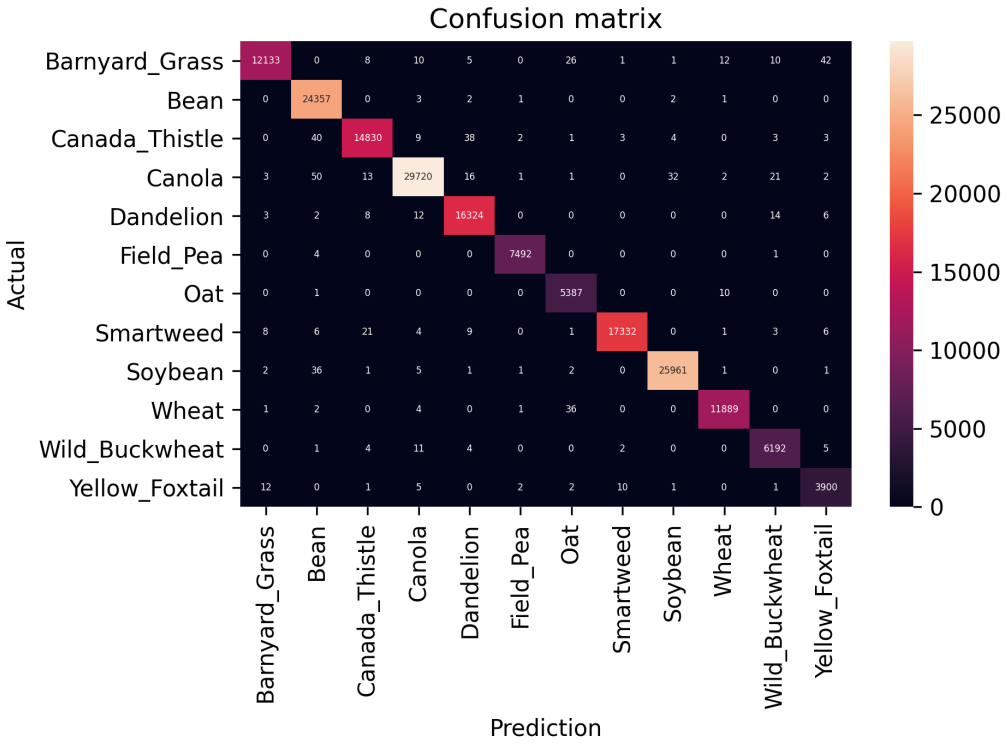

Figure 5.2 Confusion matrix of E002 model

Table 5.2 Evaluation metrics for E002 model on test dataset

	precision (%)	recall (%)	F1-score (%)	support
Barnyard_Grass	99.76	99.06	99.41	12248
Bean	99.42	99.96	99.69	24366
Canada_Thistle	99.62	99.31	99.47	14933
Canola	99.79	99.53	99.66	29861
Dandelion	99.54	99.73	99.63	16369
Field_Pea	99.89	99.93	99.91	7497
Oat	98.74	99.80	99.26	5398
Smartweed	99.91	99.66	99.78	17391
Soybean	99.85	99.81	99.83	26011
Wheat	99.77	99.63	99.70	11933
Wild_Buckwheat	99.15	99.57	99.36	6219
Yellow_Foxtail	98.36	99.14	98.75	3934
accuracy			99.63	176160
macro avg	99.48	99.59	99.54	176160
weighted avg	99.64	99.63	99.64	176160

5.2.3 Experiment E003 (3 convolution blocks and 3x3 filters)

In this experiment, the model is designed with 3 convolution blocks and stride size of 3×3 for horizontal and vertical movements, respectively. Then a flattening layer is used, followed by 1 fully connected layer (1024) with ReLU activation, 1 dropout layer (0.1), and a final fully connected layer with a softmax activation. The training batch size is 64, and the maximum epochs allowed is 45. The Adam optimizer is used for training with a learning rate of 0.001. The difference between E002 and E003 is the number of convolution blocks. The training and validation accuracy and loss of the model while training is included in Appendix A. The models are tested on the test dataset, the confusion matrix is shown in Fig. 5.3, and the performance metrics are shown in Table 5.3. The overall accuracy of this experiment on all classes is 99.74%. The micro-avg precision and recall of classification for all classes is 99.79% and 99.68% for all classes respectively. The micro-avg F1-score of all type of test images is 99.70%. The results of Yellow-Foxtail and Oat classification using E003 model are 99.72% and 99.47% respectively which are higher than the result of the previous model (E002). It shows the E003 outperforms distinguishing unrecognizable plant images like Yellow-Foxtail and Oat plant images.

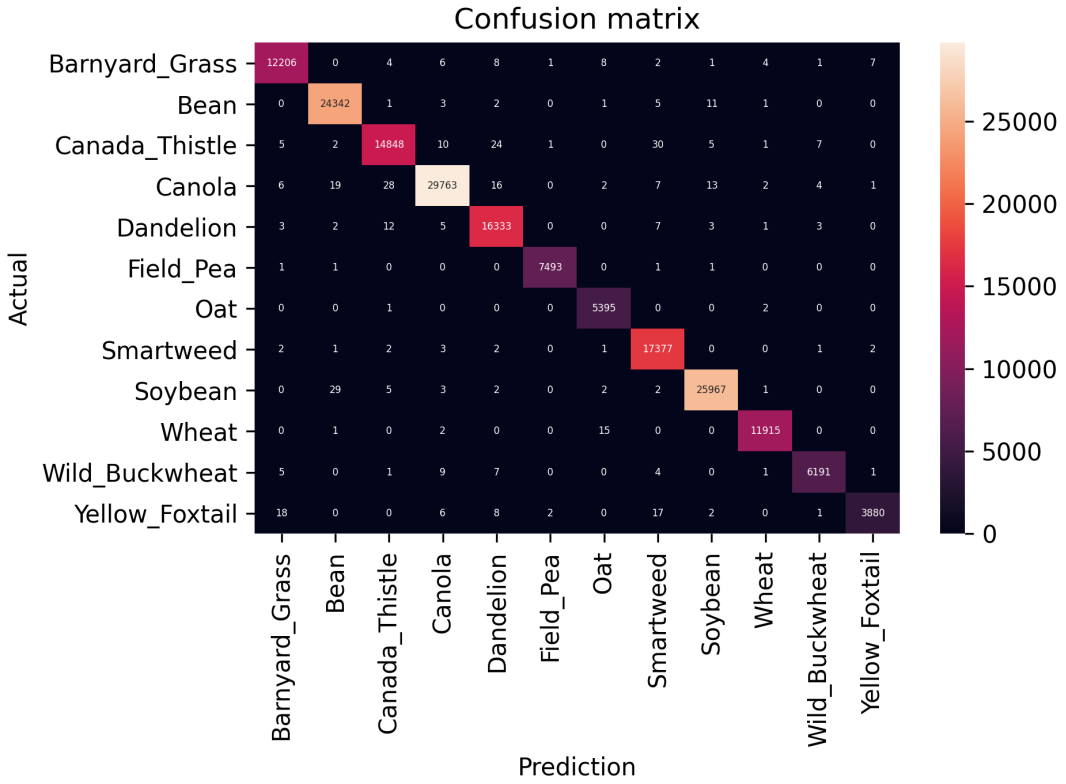

Figure 5.3 Confusion matrix of E003 model

Table 5.3 Evaluation metrics for E003 model on test dataset

	precision (%)	recall (%)	F1-score (%)	support
Barnyard_Grass	99.67	99.66	99.67	12248
Bean	99.77	99.90	99.84	24366
Canada_Thistle	99.64	99.43	99.53	14933
Canola	99.84	99.67	99.76	29861
Dandelion	99.58	99.78	99.68	16369
Field_Pea	99.95	99.95	99.95	7497
Oat	99.47	99.94	99.70	5398
Smartweed	99.57	99.92	99.74	17391
Soybean	99.86	99.83	99.85	26011
Wheat	99.89	99.85	99.87	11933
Wild_Buckwheat	99.73	99.55	99.64	6219
Yellow_Foxtail	99.72	98.63	99.17	3934
accuracy			99.74	176160
macro avg	99.72	99.68	99.70	176160
weighted avg	99.74	99.74	99.74	176160

5.2.4 Experiment E004 (3 convolution blocks and 5x5 filters)

Experiment E004 is similar to experiment E003 described at Section 5.2.3. The difference is that in the convolutional layers, instead of using strides of 3 and 3, the strides are 5 and 5. The plot of the accuracy and loss of the model is shown in Fig. A.7 and Fig. A.8, while the confusion matrix and performance metrics are shown in Fig. 5.4 and Table 5.4.

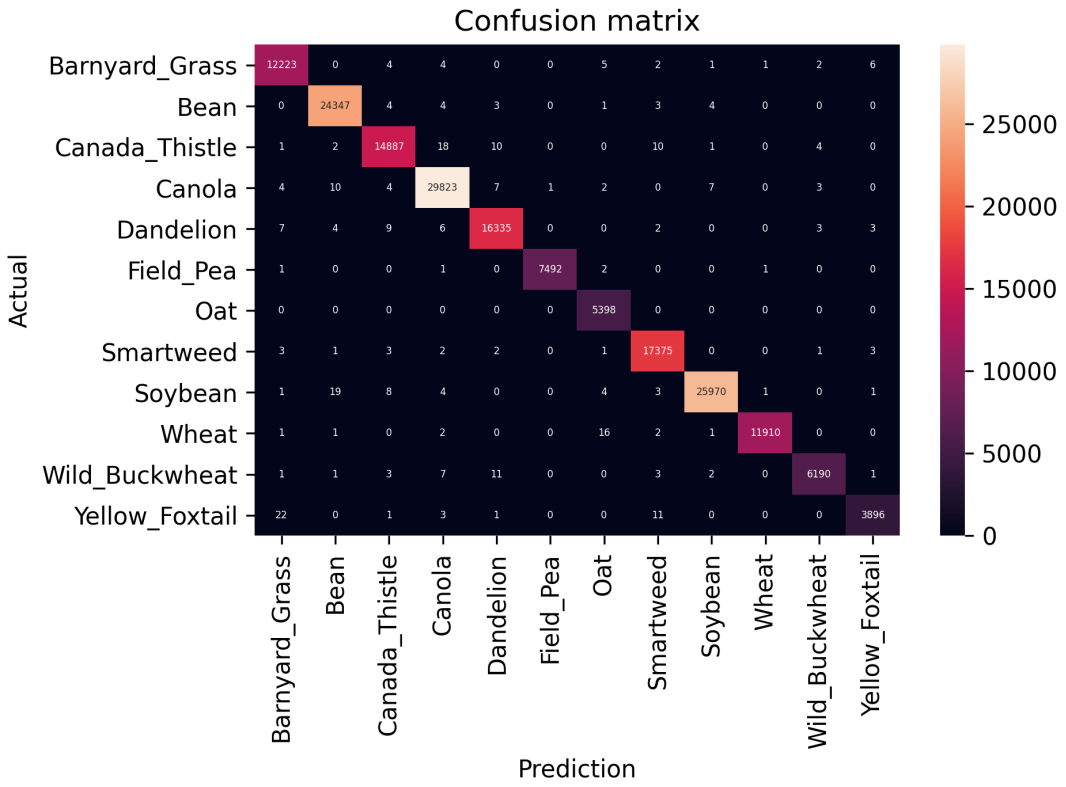

Figure 5.4 Confusion matrix of E004 model

Table 5.4 Evaluation metrics for E004 model on test dataset

	precision (%)	recall (%)	F1-score (%)	support
Barnyard_Grass	99.67	99.80	99.73	12248
Bean	99.84	99.92	99.88	24366
Canada_Thistle	99.76	99.69	99.73	14933
Canola	99.83	99.87	99.85	29861
Dandelion	99.79	99.79	99.79	16369
Field_Pea	99.99	99.93	99.96	7497
Oat	99.43	100.00	99.71	5398
Smartweed	99.79	99.91	99.85	17391
Soybean	99.94	99.84	99.89	26011
Wheat	99.97	99.81	99.89	11933
Wild_Buckwheat	99.79	99.53	99.66	6219
Yellow_Foxtail	99.64	99.03	99.34	3934
accuracy			99.82	176160
macro avg	99.79	99.76	99.77	176160
weighted avg	99.82	99.82	99.82	176160

5.2.5 Experiment E010 (Tree of CNN with Transfer Learning)

In this experiment, we use the tree of CNN structure to create a sophisticated model (see Section 4.4.3). The tree of CNN model is a combination of the following CNNs:

- A VGG16 network that is pre-trained on ImageNet dataset, and the pre-trained weights are frozen during training. This network is identical to the VGG16 network used in experiment E001. Only the last layer is trained on the whole training dataset (as described in Section 4.4.1) with 3 target groups. The purpose of this network is to classify an image into one of the target groups (G1, G2, or G3).
- Three separate VGG16 networks identical to the VGG16 network used in experiment E001, but trained on the training dataset of each target group. These networks are used to classify an image into the final classes (labels) listed in Table 4.4.

Overall, four separate networks are trained individually, each on a separate dataset. One generalist network to classify an input image into groups and three specialist networks to classify an input image into the final classes. A controller class is coded to supervise the process of handling each image and passing it through the correct network until the final classification is obtained. In general, the training networks on the dataset are not end-to-end since we have four different CNN groups. Plots of the accuracy and loss of the model, confusion matrix, and performance metrics for each separate network are given in Appendix A. The final confusion matrix is shown in Fig 5.5 and Table 5.5 shows the performance metrics.

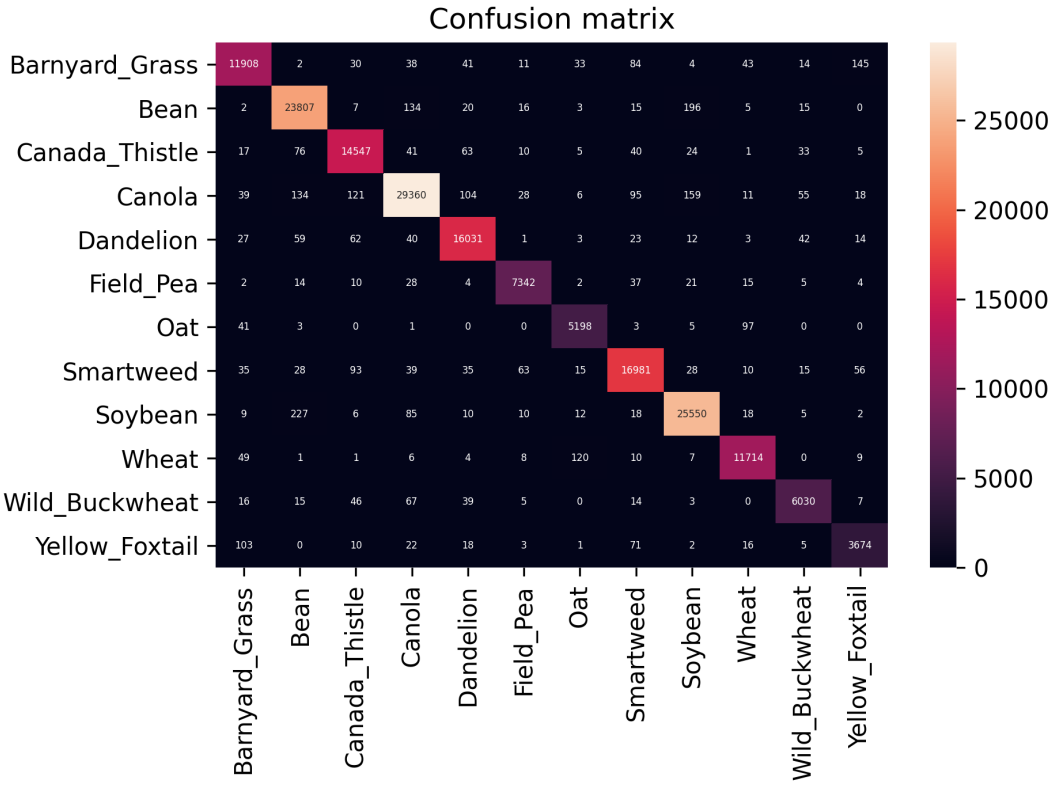

Figure 5.5 Confusion matrix of E010 model

Table 5.5 Evaluation metrics for E010 model on test dataset

	precision (%)	recall (%)	F1-score (%)	support
Barnyard_Grass	96.40	97.22	96.81	12248
Bean	98.29	97.71	98.00	24366
Canada_Thistle	97.88	97.42	97.65	14933
Canola	97.44	98.32	97.88	29861
Dandelion	98.25	97.94	98.09	16369
Field_Pea	98.10	97.93	98.02	7497
Oat	97.20	96.29	96.74	5398
Smartweed	97.60	97.64	97.62	17391
Soybean	98.45	98.23	98.34	26011
Wheat	98.20	98.16	98.18	11933
Wild_Buckwheat	96.60	96.96	96.78	6219
Yellow_Foxtail	93.61	93.39	93.50	3934
accuracy			97.72	176160
macro avg	97.34	97.27	97.30	176160
weighted avg	97.72	97.72	97.72	176160

5.2.6 Experiment E011 (Tree of CNN with simple CNN)

This experiment is similar to experiment E010, and the difference is that the CNN models are not pre-trained. Instead, this experiment uses the model used in the E002 experiment. That is, the model architecture used in all the CNNs is the simple CNN described in Section 4.4.2 with 5 convolutional blocks. Four networks are trained individually, each on separate datasets: One generalist network to classify an input image into groups (G1,G2, and G3) and 3 specialist networks to classify an input image into the final classes. For the end-to-end evaluation, a controller is coded to supervise the process of handling each image and passing it through the correct network until the final classification is obtained. The history, confusion matrix, and performance metrics for each separate network can be found in Appendix A. The final confusion matrix is shown in Fig. 5.6 and Table 5.6 shows the performance metrics.

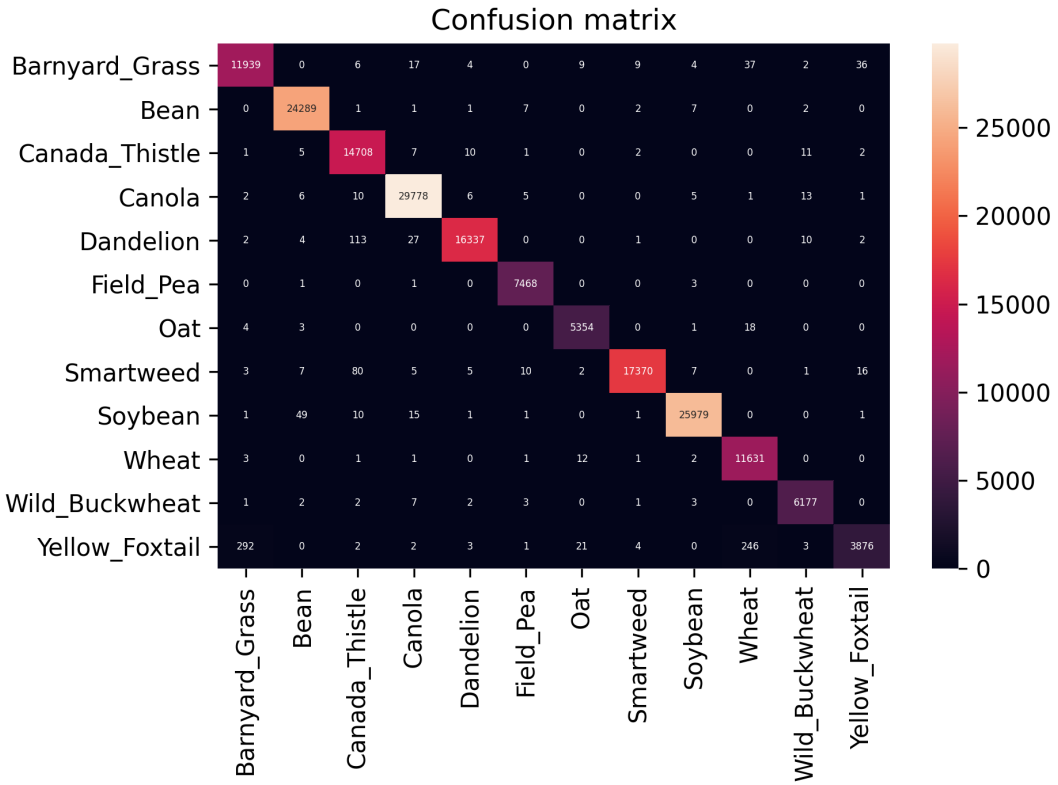

Figure 5.6 Confusion matrix of E011 model

Table 5.6 Evaluation metrics for E011 model on test dataset

	precision (%)	recall (%)	F1-score (%)	support
Barnyard_Grass	98.97	97.48	98.22	12248
Bean	99.91	99.68	99.80	24366
Canada_Thistle	99.74	98.49	99.11	14933
Canola	99.84	99.72	99.78	29861
Dandelion	99.04	99.80	99.42	16369
Field_Pea	99.93	99.61	99.77	7497
Oat	99.52	99.18	99.35	5398
Smartweed	99.22	99.88	99.55	17391
Soybean	99.70	99.88	99.79	26011
Wheat	99.82	97.47	98.63	11933
Wild_Buckwheat	99.66	99.32	99.49	6219
Yellow_Foxtail	87.10	98.53	92.46	3934
accuracy			99.29	176160
macro avg	98.54	99.09	98.78	176160
weighted avg	99.33	99.29	99.30	176160

5.2.7 Experiment E012 (Tree of CNN with simple CNN)

This experiment is similar to experiment E010, and the difference is that the CNN models are not pre-trained. Instead, this experiment uses the model used in the E003 experiment. That is, the model architecture used in all the CNNs is the simple CNN described in Section 4.4.2 with 3 convolutional blocks. Four networks are trained individually, each on separate datasets: One generalist network to classify an input image into groups (G1, G2, and G3) and 3 specialist networks to classify an input image into the final classes. For the end-to-end evaluation, a controller is coded to supervise the process of handling each image and passing it through the correct network until the final classification is obtained. The history, confusion matrix, and performance metrics for each separate network can be found in Appendix A. The final confusion matrix is shown in Fig. 5.7 and Table 5.7 shows the performance metrics.

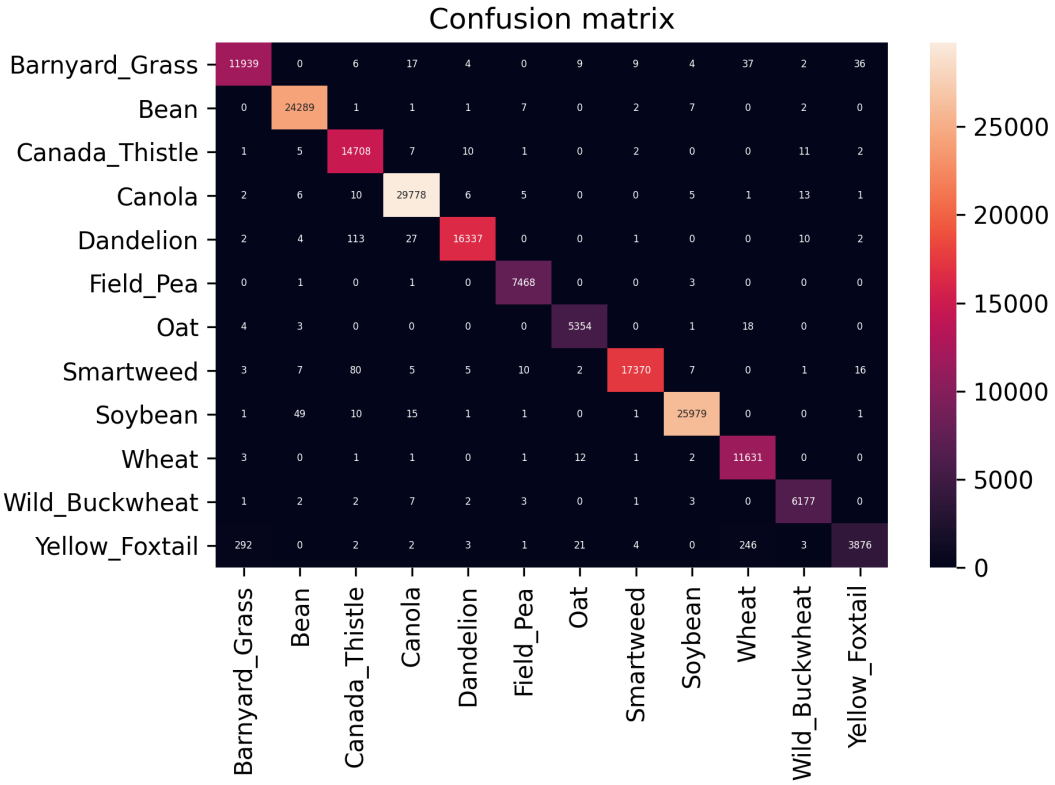

Figure 5.7 Confusion matrix of E012 model

Table 5.7 Evaluation metrics for E012 model on test dataset

	precision (%)	recall (%)	F1-score (%)	support
Barnyard_Grass	99.24	99.60	99.42	12248
Bean	99.92	99.49	99.70	24366
Canada_Thistle	99.54	99.46	99.50	14933
Canola	99.68	99.77	99.73	29861
Dandelion	99.41	99.70	99.55	16369
Field_Pea	99.79	99.80	99.79	7497
Oat	98.34	99.91	99.12	5398
Smartweed	99.79	99.69	99.74	17391
Soybean	99.65	99.90	99.77	26011
Wheat	99.97	99.15	99.56	11933
Wild_Buckwheat	99.25	99.57	99.41	6219
Yellow_Foxtail	99.41	97.92	98.66	3934
accuracy			99.61	176160
macro avg	99.50	99.50	99.50	176160
weighted avg	99.61	99.61	99.61	176160

5.3 Test Dataset Evaluation

All experiments introduced in Section 5.2 were trained on the training dataset and assessed on the test dataset, which is explained in Section 4. The test dataset contains 176,615 images, including 12 different classes, namely, Wild Buckwheat, Barnyard Grass, Canada Thistle, Canola, Dandelion, Smartweed, Bean, Soybean, Wheat, Oat, Field Pea, and Yellow Foxtail. The test dataset was kept constant for all experiments. We called this test dataset the Main test dataset. Section 5.2 shows the accurate classification of the test dataset on seven CNN models. Table 5.8 shows the name of CNN models with their results studied in this thesis.

In addition to the Main test dataset, we took another 3 different test data sets prepared by Beck et al. [12] to further evaluate the results. These test datasets contain 7 different classes, including Barnyard Grass, Canada Thistle, Canola, Dandelion, Smartweed, Wild Buckwheat, and Yellow Foxtail. Since these 3 test sets used Sightline to evaluate their CNN model with the TerraByte data, we chose these test datasets to first evaluate our models and compare our results to the Sightline results. The first dataset recorded as "Same Angles" includes 3,494 images which are similar to the Main test dataset in terms of the background (blue background) and the camera angles. The second dataset, named "Random Angles" includes 520 images. The pictures were taken from different angles. The plant images have a blue background and are taken with different random angles as well as the Main test set. The "Smart Phone" dataset includes 56 plant images with diverse backgrounds and angles of the Main test dataset. Data preprocessing has been done on these 3 datasets to make them ready to use as the test datasets for all the defined seven experiment models in Section 5.2. We applied the following 2 preprocessing operations. First, the images were re-labelled to have similar labels as the Main test dataset. Second, a labelling spreadsheet (CSV file) was created based on the filename of plants and their labels. After the preprocessing step, we evaluated the seven CNN models by using the 3 new test sets: Same Angles, Random Angles and Smart Phone test datasets. We evaluated the performance of the models in two different ways. For the first approach, we used our models, which are trained on not 7 but 12 classes, since the 12 classes that our models are trained on include the seven classes in the new test datasets. The results are reported in Table 5.9. The second approach was to train models from scratch. We used the same datasets described in Section 4.2 as the baseline. Then, we filtered out the images belonging to the extra 5 classes. The remaining samples are used for training and validation. The models used for E001 and E002 experiments are slightly modified. Only the last layer is modified to have 7 classes instead of 12 classes. The testing results are reported in Tables A.10 and A.11.

The results of a model trained on the datasets [12] by Sightline, along with the accuracy of the models developed in this study, are listed in Table 5.9.

Table 5.8 Summary of eveluation for the seven CNN models on the main test dataset

Model	Accuracy (%)
E001	97.86
E002	99.63
E003	99.74
E004	99.82
E010	97.72
E011	99.29
E012	99.61

Table 5.9 Comparision of results with Sightline model

Model	Dataset	Accuracy (%)
Sightline	Same Angles	45.65
	Random Angles	61.15
	Smartphone	58.93
E001	Same Angles	49.62
	Random Angles	44.61
	Smartphone	12.50
E002	Same Angles	54.23
	Random Angles	44.23
	Smartphone	14.28
E001-sl	Same Angles	54.75
	Random Angles	51.15
	Smartphone	23.21
E002-sl	Same Angles	48.65
	Random Angles	40.77
	Smartphone	17.86
E010 Generalist	Same Angles	78.53
	Random Angles	70.19
	Smartphone	60.71
E010	Same Angles	48.94
	Random Angles	41.35
	Smartphone	12.50
E011 Generalist	Same Angles	85.26
	Random Angles	88.65
	Smartphone	55.36
E011	Same Angles	64.45
	Random Angles	52.12
	Smartphone	10.71
E012 Generalist	Same Angles	74.30
	Random Angles	76.15
	Smartphone	57.14
E012	Same Angles	57.18
	Random Angles	45.96
	Smartphone	12.50

5.4 Peer Comparison and Analysis

Table 5.8 shows the result of all 7 CNN models on the Main test set that all are similarly high. It shows the simple CNN models, including E002 with 99.63% accuracy, E003 with 99.74%, and E004 with 99.82% accuracy have better results compared with 97.86% accuracy for the transfer learning with the VGG16 model. The generalist-specialist model used simple CNN architecture with 99.61% accuracy work significantly better than using transfer learning with the VGG16 model with 97.72% accuracy. Based on Table 5.9, it can be seen that the accuracy of our models on the Same Angles and Random Angles dataset are similar except that the Sightline model has better accuracy on the Smartphone dataset. Classification accuracy of the E011 and E012 models on the Same Angles dataset is higher than the Sightline result. The accuracy of classification of the generalist CNN of the E010, E011, and E012 are also shown. It can be seen that on the generalist level (which classifies the input image into G1, G2, or G3) is relatively high, which is expected due to our CNN design to classify the similar plant images. The E011 and E012 models show slightly higher accuracies over E001 and E002 models.

We can consider several reasons why the results of our models E001 to E012 CNN, on the Main test dataset, are of higher accuracy than the Random Angles and Smart Phone test datasets.

Firstly, before training our models, we filtered the training and test datasets by the age of each class of plants. The networks are trained with plant images at specific ages. For example, Canola images range from 14 to 65 days old (Section 4 introduced the applied age filter on the images). It means the networks trained on Canola images between this range. There is a possibility that these new test datasets contain plant images out of the range we used in the training dataset.

Secondly, since the images Sightline picked for their test sets, taken on 27 and 29 April are a very small portion of all images from the TerraByte dataset that we picked for training and test sets, we can not evaluate our models accurately by just test sets from a small proportion of data. For instance, [13] shows the number of images that have been taken in April is 15,015, which is a very small portion of 1.2 million images from the TerraByte dataset.

Generalization and overfitting problems might be the third reason. If our models were better in generalization, we might have as good accuracy as on these new test datasets. When the result of the test dataset is far from the validation dataset, it might be due to overfitting. But in our project, we cannot have this problem because of the following reasons. A model can overfit the training dataset when there are not enough variations of training data. When data is not sufficient in number, the model can not generalize well. The overfitting issue can be solved by providing more data for the model to train. Our models are trained

with 530,000 samples, and the reason for overfitting does not seem to be a lack of data. Another reason that could contribute to overfitting is the complexity of the model. A complex model with many layers and parameters can overfit the data and needs more training data to generalize well. Our models are relatively simple, and we have used dropout layers to reduce overfitting. Overfitting is detected when the model does not generalize well. We use a validation dataset to help the model generalize in the training process. The model does not train on the validation dataset. At the end of each epoch, the model is evaluated by running the model against the validation dataset. When the model shows high accuracy on the training dataset and low accuracy on the validation dataset, it means overfitting occurred. Overfitting (and underfitting) can be visualized by plotting the history of the accuracy and loss over training time (epoch). When accuracy curves for training and validation diverge, it is a sign of overfitting or underfitting. Our models, while training, did not show such behaviour. A model can also overfit if the validation dataset is not helpful. A validation dataset that does not have enough samples for different classes can affect the model. For this research, the whole dataset is randomized before splitting into training, validation, and test sets. Also, as discussed in Section 4.2, we have used 60% of the dataset for training, 20% for validation, and 20% for testing. Also, we have made sure that the validation and testing datasets have the same ratio of classes as the training dataset (See Figure 4.6 for more details). For testing purposes, we have used the test dataset. The test dataset is not used for training or validation. All the explored models in this thesis perform pretty well during training and do not show overfitting signs. However, they might not be generalizing well on the test datasets. We suspect that one reason could be that the samples in the validation dataset, although randomized, are similar to the ones in the training dataset, in a way that the validation dataset is not helpful and the models might be overfitting. That could explain why the accuracy of the models is high on the test dataset prepared by this research, but the models perform poorly on the foreign dataset described above. In a future study, we could apply several generalization methods, like data augmentation and change the background of the pictures on the validation dataset.

Chapter 6

Conclusion

The automatic classification of crops and weeds is challenging in computer vision due to similar shape and colour representation across species. We aimed to design and implement different CNN architectures to solve the plant classification problems for similar plant images. We used a large plant dataset containing 1,207,829 labelled images from the TerraByte dataset, captured and produced from 2020-04-01 to 2020-12-13. This dataset includes 12 plant species, including wheat, canola, oat, bean, soybean, field pea, yellow foxtail, dandelion, wild bucketwheat, barnyard grass, smartweed, and Canada thistle. Most of these species are very challenging to be recognized by humans due to the similar size and shape of their leaves. Before implementing CNN models, the following three data preprocessing operations were applied to the plant images. Firstly, image re-labelling was done to simplify the problem space. Secondly, the plants were filtered by age. For example, we only kept images of canola with ages between 14 to 65 days. Thirdly, plant species were placed into similar groups. Then, three diverse CNN architectures were explored and implemented to solve the classification of highly similar plant images. One of those was the transfer learning using VGG-16, that could classify the plant species with 97.86% accuracy. The second CNN architecture was related to the simple CNN models, including different layers and parameters. The highest accuracy for those models was 99.82%. The hierarchical tree-based CNN (general-specialist networks) was the third architecture explored and implemented. The generalist-specialist networks could improve transfer learning using VGG-16 from 97.86% accuracy to 99.61% accuracy. The performance of each mentioned CNN architecture was evaluated and compared by randomly picking a test set from the database. Additionally, we assessed and compared our CNN models with a model produced by Sightline Innovation, which worked with a portion of the TerraByte dataset. The results showed that the generalist-specialist networks had a better performance than the Sightline model on the same plant images by 18.80%.

This thesis explored CNN architectures to solve highly similar plant species using the TerraByte dataset. Even so, we can mention below suggestions for future work:

- Since we got different results from the same dataset with different time duration, one future task can be research into important parameters for the appearance of plant images, like the effect of temperature, sunshine or the lighting system on the greenhouse for different months of the year.
- Image segmentation to separate the plant from the background can be employed as a preprocessing step. This will eliminate the possibility that the models would account for the background.
- Research into classifying breed of plants. In the Terrabyte dataset, the breed of plants is available. This can be used to investigate different breeds of plants.
- Creating a dataset from images taken from field and greenhouse would build a foundation for further research to classify real-world images. These images can be used as validation and test datasets in an effort to create and train models that can generalize what they learn to field images.
- Explore multispectral and hyperspectral imaging techniques to determine if the information present in the additional spectral bands improves the classification metrics or allows the development of models with fewer training samples.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Ahmed, K., Baig, M. H., and Torresani, L. (2016). Network of experts for large-scale image categorization. In *European Conference on Computer Vision*, pages 516–532. Springer.

[3] Aktar, M. W., Paramasivam, M., Sengupta, D., Purkait, S., Ganguly, M., and Banerjee, S. (2009). Impact assessment of pesticide residues in fish of Ganga river around Kolkata in West Bengal. *Environmental monitoring and assessment*, 157(1):97–104.

[4] Amara, J., Bouaziz, B., and Algergawy, A. (2017). A deep learning-based approach for banana leaf diseases classification. *Datenbanksysteme für Business, Technologie und Web (BTW 2017)-Workshopband*.

[5] Bai, K. (2019). A comprehensive introduction to different types of convolutions in deep learning. *Towards data science. URL: https://towardsdatascience. com/acomprehensive-introduction-to-different-types-of-convolutions-in-deeplearning-669281e58215*.

[6] Balaban, S. (2015). Deep learning and face recognition: the state of the art. In *Biometric and surveillance technology for human and activity identification XII*, volume 9457, page 94570B. International Society for Optics and Photonics.

[7] Baldi, P. and Sadowski, P. J. (2013). Understanding dropout. *Advances in neural information processing systems*, 26:2814–2822.

[8] Banerjee, C., Mukherjee, T., and Pasiliao Jr, E. (2019). An empirical study on generalizations of the ReLU activation function. In *Proceedings of the 2019 ACM Southeast Conference*, pages 164–167.

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

[10] Barré, P., Stöver, B. C., Müller, K. F., and Steinhage, V. (2017). Leafnet: A computer vision system for automatic plant species identification. *Ecological Informatics*, 40:50–56.

[11] Baweja, P., Kumar, S., and Kumar, G. (2020). Fertilizers and pesticides: Their impact on soil health and environment. In *Soil Health*, pages 265–285. Springer.

[12] Beck, M. A., Liu, C.-Y., Bidinosti, C. P., Henry, C. J., Godee, C. M., and Ajmani, M. (2020). An embedded system for the automated generation of labeled plant images to enable machine learning applications in agriculture. *Plos One*, 15(12):e0243923.

[13] Beck, M. A., Liu, C.-Y., Bidinosti, C. P., Henry, C. J., Godee, C. M., and Ajmani, M. (2021). Presenting an extensive lab-and field-image dataset of crops and weeds for computer vision tasks in agriculture. *arXiv preprint arXiv:2108.05789*.

[14] Bergerman, M., Billingsley, J., Reid, J., and van Henten, E. (2016). Robotics in agriculture and forestry. In *Springer handbook of robotics*, pages 1463–1492. Springer.

[15] Bongiovanni, R. and Lowenberg-DeBoer, J. (2004). Precision agriculture and sustainability. *Precision agriculture*, 5(4):359–387.

[16] Calicioglu, O., Flammini, A., Bracco, S., Bellù, L., and Sims, R. (2019). The future challenges of food and agriculture: An integrated analysis of trends and solutions. *Sustainability*, 11(1):222.

[17] Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2017). Yale-CMU-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268.

[18] Camgözlü, Y. and Kutlu, Y. (2020). Analysis of filter size effect in deep learning. *arXiv preprint arXiv:2101.01115*.

[19] Camou-Guerrero, A., Reyes-García, V., Martínez-Ramos, M., and Casas, A. (2008). Knowledge and use value of plant species in a Rarámuri community: a gender perspective for conservation. *Human ecology*, 36(2):259–272.

[20] Chicco, D. and Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13.

[21] Christlein, V., Spranger, L., Seuret, M., Nicolaou, A., Král, P., and Maier, A. (2019). Deep generalized max pooling. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1090–1096. IEEE.

[22] Cover, T. M. (1999). *Elements of information theory*. John Wiley & Sons.

[23] Cox, D. D. and Dean, T. (2014). Neural networks and neuroscience-inspired computer vision. *Current Biology*, 24(18):R921–R929.

[24] Coyne, D. L. (2007). *Practical plant nematology: a field and laboratory guide*. IITA.

[25] Cunningham, P., Cord, M., and Delany, S. J. (2008). Supervised learning. In *Machine learning techniques for multimedia*, pages 21–49. Springer.

[26] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE.

[27] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

[28] Dumoulin, V., Visin, F., and Box, G. (2018a). A guide to convolution arithmetic for deep learning. arxiv prepr. *arXiv preprint arXiv:1603.07285*.

[29] Dumoulin, V., Visin, F., and George, E. (2018b). Box. a guide to convolution arithmetic for deep learning. Technical report, Tech. rep. 2018. url: http://ethanschoonover. com/solarized.

[30] El Housseini, A., Toumi, A., and Khenchaf, A. (2017). Deep learning for target recognition from SAR images. In *2017 Seminar on Detection Systems Architectures and Technologies (DAT)*, pages 1–5. IEEE.

[31] Elz, R. and Bush, R. (1996). Serial number arithmetic. Technical report, RFC 1982, August.

[32] Galli, V., Nixon, C. C., Strbo, N., Artesi, M., de Castro-Amarante, M. F., McKinnon, K., Fujikawa, D., Omsland, M., Washington-Parks, R., Romero, L., et al. (2019). Essential role of human T cell leukemia virus type 1 orf-I in lethal proliferation of CD4+ cells in humanized mice. *Journal of virology*, 93(19):e00565–19.

[33] Gao, H., Wang, Z., and Ji, S. (2018). Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424.

[34] García, S., Luengo, J., and Herrera, F. (2015). *Data preprocessing in data mining*, volume 72. Springer.

[35] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., and Garcia-Rodriguez, J. (2017). A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*.

[36] Glandberger, O. and Fredriksson, D. (2020). Neural network regularization for generalized heart arrhythmia classification.

[37] Gonzalez-Diaz, I., Benois-Pineau, J., Domenger, J.-P., Cattaert, D., and de Rugy, A. (2019). Perceptually-guided deep neural networks for ego-action prediction: Object grasping. *Pattern Recognition*, 88:223–235.

[38] Grandini, M., Bagli, E., and Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

[39] Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., et al. (2020). GluonCV and GluonNLP: Deep learning in computer vision and natural language processing. *J. Mach. Learn. Res.*, 21(23):1–7.

[40] Guo, T., Dong, J., Li, H., and Gao, Y. (2017). Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724. IEEE.

[41] Hagiwara, K. and Fukumizu, K. (2008). Relation between weight size and degree of over-fitting in neural network regression. *Neural networks*, 21(1):48–58.

[42] Hasan, M. S. et al. (2017). An application of pre-trained CNN for image classification. In *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–6. IEEE.

[43] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[44] Howard, A. G. (2013). Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*.

[45] Hunter, M. C., Smith, R. G., Schipanski, M. E., Atwood, L. W., and Mortensen, D. A. (2017). Agriculture in 2050: recalibrating targets for sustainable intensification. *Bioscience*, 67(4):386–391.

[46] Jia, W., Tian, Y., Luo, R., Zhang, Z., Lian, J., and Zheng, Y. (2020). Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot. *Computers and Electronics in Agriculture*, 172:105380.

[47] Jiang, H., Zhang, C., Qiao, Y., Zhang, Z., Zhang, W., and Song, C. (2020). CNN feature based graph convolutional network for weed and crop recognition in smart farming. *Computers and Electronics in Agriculture*, 174:105450.

[48] Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

[49] Kamilaris, A. and Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90.

[50] Kartikeyan, P. and Shrivastava, G. (2021). Review on emerging trends in detection of plant diseases using image processing with machine learning. *International Journal of Computer Applications*, 975:8887.

[51] Kaya, A., Keceli, A. S., Catal, C., Yalic, H. Y., Temucin, H., and Tekinerdogan, B. (2019). Analysis of transfer learning for deep neural network based plant classification models. *Computers and electronics in agriculture*, 158:20–29.

[52] Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516.

[53] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[54] Kontschieder, P., Fiterau, M., Criminisi, A., and Bulo, S. R. (2015). Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475.

[55] Korotchenya, V. (2019). Digital agriculture and agricultural production efficiency: exploring prospects for Russia. *Revista ESPACIOS*, 40(22):22–35.

[56] Krishnakumari, K., Sivasankar, E., and Radhakrishnan, S. (2020). Hyperparameter tuning in convolutional neural networks for domain adaptation in sentiment classification (HTCNN-DASC). *Soft Computing*, 24(5):3511–3527.

[57] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

[58] Kyrkou, C., Plastiras, G., Theocharides, T., Venieris, S. I., and Bouganis, C.-S. (2018). DroNet: Efficient convolutional neural network detector for real-time UAV applications. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 967–972. IEEE.

[59] Lameski, P., Zdravevski, E., Trajkovik, V., and Kulakov, A. (2017). Weed detection dataset with RGB images taken under variable light conditions. In *International Conference on ICT Innovations*, pages 112–119. Springer.

[60] Lasseck, M. (2017). Image-based plant species identification with deep convolutional neural networks. In *CLEF (Working Notes)*.

[61] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[62] LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

[63] Lee, S. H., Chan, C. S., Wilkin, P., and Remagnino, P. (2015). Deep-plant: Plant identification with convolutional neural networks. In *2015 IEEE international conference on image processing (ICIP)*, pages 452–456. IEEE.

[64] Li, F., Karpathy, A., and Johnson, J. (2015). Cs231n: Convolutional neural networks for visual recognition. Course notes.

[65] Maksymiv, I. (2015). Pesticides: benefits and hazards. *Journal of Vasyl Stefanyk Precarpathian National University*, 2(1):70–76.

[66] Manoharan, S., Sariffodeen, B., Ramasinghe, K., Rajaratne, L., Kasthurirathna, D., and Wijekoon, J. L. (2020). Smart plant disorder identification using computer vision technology. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0445–0451. IEEE.

[67] Mao, X., Hijazi, S., Casas, R., Kaul, P., Kumar, R., and Rowen, C. (2016). Hierarchical CNN for traffic sign recognition. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 130–135. IEEE.

[68] Mayr, A., Klambauer, G., Unterthiner, T., and Hochreiter, S. (2016). Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80.

[69] Milletari, F., Ahmadi, S.-A., Kroll, C., Plate, A., Rozanski, V., Maiostre, J., Levin, J., Dietrich, O., Ertl-Wagner, B., Bötzel, K., et al. (2017). Hough-cnn: deep learning for segmentation of deep brain regions in MRI and ultrasound. *Computer Vision and Image Understanding*, 164:92–102.

[70] Min, S., Lee, B., and Yoon, S. (2017). Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869.

[71] Mohanty, S. P., Hughes, D. P., and Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419.

[72] Molina-Villa, M. A., Solaque-Guzmán, L. E., et al. (2016). Machine vision system for weed detection using image filtering in vegetables crops. *Revista Facultad de Ingeniería Universidad de Antioquia*, (80):124–130.

[73] Murthy, V. N., Maji, S., and Manmatha, R. (2015). Automatic image annotation using deep learning representations. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 603–606.

[74] Namin, S. T., Esmaeilzadeh, M., Najafi, M., Brown, T. B., and Borevitz, J. O. (2018). Deep phenotyping: deep learning for temporal phenotype/genotype classification. *Plant Methods*, 14(1):1–14.

[75] Parker, J. A., Kenyon, R. V., and Troxel, D. E. (1983). Comparison of interpolating methods for image resampling. *IEEE Transactions on medical imaging*, 2(1):31–39.

[76] Prashar, P. and Shah, S. (2016). Impact of fertilizers and pesticides on soil microflora in agriculture. In *Sustainable agriculture reviews*, pages 331–361. Springer.

[77] Ramprasath, M., Anand, M. V., and Hariharan, S. (2018). Image classification using convolutional neural networks. *International Journal of Pure and Applied Mathematics*, 119(17):1307–1319.

[78] Rehman, T. U., Mahmud, M. S., Chang, Y. K., Jin, J., and Shin, J. (2019). Current and future applications of statistical machine learning algorithms for agricultural machine vision systems. *Computers and electronics in agriculture*, 156:585–605.

[79] Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265–278.

[80] Rojas, R. (1996). The backpropagation algorithm. In *Neural networks*, pages 149–182. Springer.

[81] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

[82] Roy, D., Panda, P., and Roy, K. (2020). Tree-CNN: a hierarchical deep convolutional neural network for incremental learning. *Neural Networks*, 121:148–160.

[83] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[84] Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., and McCool, C. (2016). Deepfruits: A fruit detection system using deep neural networks. *Sensors*, 16(8):1222.

[85] Saiz-Rubio, V. and Rovira-Más, F. (2020). From smart farming towards agriculture 5.0: A review on crop data management. *Agronomy*, 10(2):207.

[86] Sarwar, S. S., Ankit, A., and Roy, K. (2019). Incremental learning in deep convolutional neural networks using partial network sharing. *IEEE Access*, 8:4615–4628.

[87] Sarwar, S. S., Panda, P., and Roy, K. (2017). Gabor filter assisted energy efficient fast learning convolutional neural networks. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE.

[88] Shen, S., Basist, A., and Howard, A. (2010). Structure of a digital agriculture system and agricultural risks due to climate changes. *Agriculture and Agricultural Science Procedia*, 1:42–51.

[89] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[90] Singh, P., Raj, P., and Namboodiri, V. P. (2020). Eds pooling layer. *Image and Vision Computing*, 98:103923.

[91] Sinha, R. K., Pandey, R., and Pattnaik, R. (2018). Deep learning for computer vision tasks: a review. *arXiv preprint arXiv:1804.03928*.

[92] Sinha, T., Verma, B., and Haidar, A. (2017). Optimization of convolutional neural network parameters for image classification. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE.

[93] Sommer, F., Fouss, F., and Saerens, M. (2017). Modularity-driven kernel k-means for community detection. In *International Conference on Artificial Neural Networks*, pages 423–433. Springer.

[94] Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. (2022). Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*.

[95] Srivastava, N. and Salakhutdinov, R. (2013). Discriminative transfer learning with tree-based priors. In *NIPS*, page 8. Citeseer.

[96] Strassemeyer, J., Daehmlow, D., Dominic, A., Lorenz, S., and Golla, B. (2017). SYNOPS-WEB, an online tool for environmental risk assessment to evaluate pesticide strategies on field level. *Crop protection*, 97:28–44.

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

[98] Tammina, S. (2019). Transfer learning using vgg-16 with deep convolutional neural network for classifying images. *International Journal of Scientific and Research Publications (IJSRP)*, 9(10):143–150.

[99] Taylor, L. and Nitschke, G. (2018). Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547. IEEE.

[100] Tieleman, T., Hinton, G., et al. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[101] Tousch, A.-M., Herbin, S., and Audibert, J.-Y. (2012). Semantic hierarchies for image annotation: A survey. *Pattern Recognition*, 45(1):333–345.

[102] Tsai, Y.-Y., Chen, P.-Y., and Ho, T.-Y. (2020). Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. In *International Conference on Machine Learning*, pages 9614–9624. PMLR.

[103] Tsenov, R.-M. (2021). *Improving LULC Map Production via Semantic Segmentation and Unsupervised Domain Adaptation*. PhD thesis, University of Winnipeg.

[104] Tudi, M., Daniel Ruan, H., Wang, L., Lyu, J., Sadler, R., Connell, D., Chu, C., and Phung, D. T. (2021). Agriculture development, pesticide application and its impact on the environment. *International Journal of Environmental Research and Public Health*, 18(3):1112.

[105] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

[106] Wang, X., Wang, H., Niu, S., and Zhang, J. (2019). Detection and localization of image forgeries using improved mask regional convolutional neural network. *Mathematical Biosciences and Engineering*, 16(5):4581–4593.

[107] Weigend, A. (1994). On overfitting and the effective number of hidden units. In *Proceedings of the 1993 connectionist models summer school*, volume 1, pages 335–342.

[108] Yan, Z., Zhang, H., Piramuthu, R., Jagadeesh, V., DeCoste, D., Di, W., and Yu, Y. (2014). HD-CNN: Hierarchical deep convolutional neural network for large scale visual recognition. *arXiv preprint arXiv:1410.0736*.

[109] Yan, Z., Zhang, H., Piramuthu, R., Jagadeesh, V., DeCoste, D., Di, W., and Yu, Y. (2015). HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 2740–2748.

[110] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*.

[111] Yu, Y., Zhang, K., Yang, L., and Zhang, D. (2019). Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Computers and Electronics in Agriculture*, 163:104846.

Appendix A

Experiment Results

A.1 Training History for E001

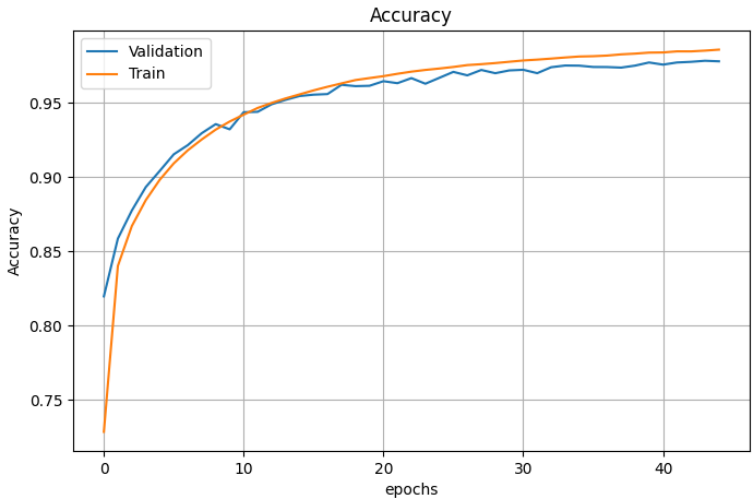

Figure A.1 Accuracy of E001 model while training

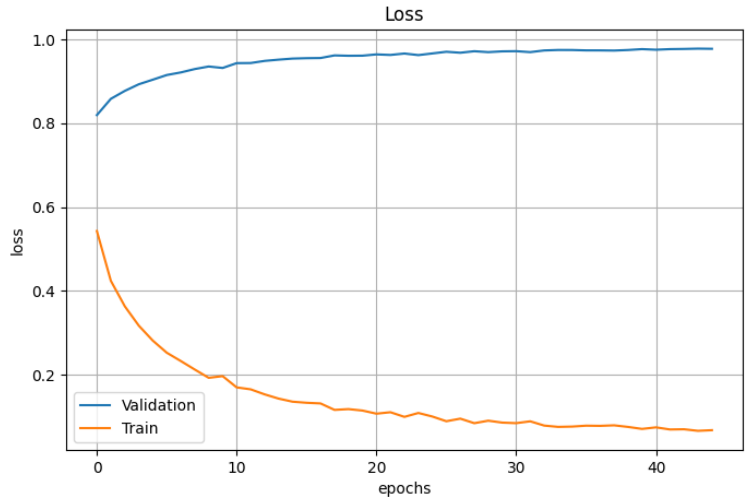

Figure A.2 Loss of E001 model while training

A.2 Training History for E002

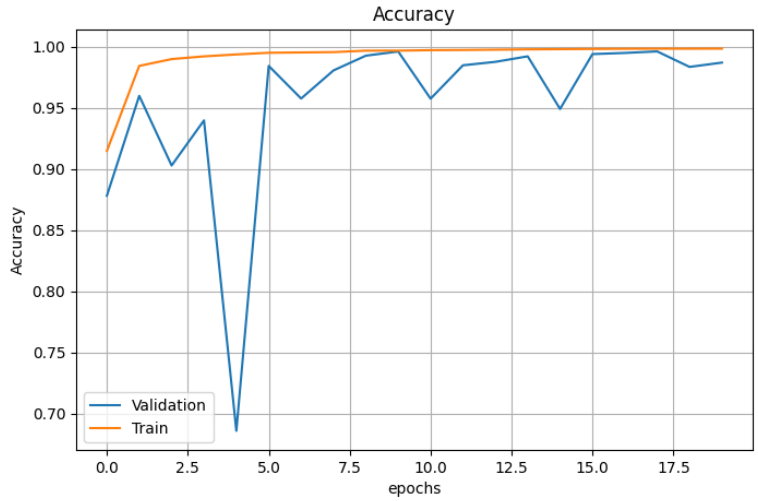

Figure A.3 Accuracy of E002 model while training

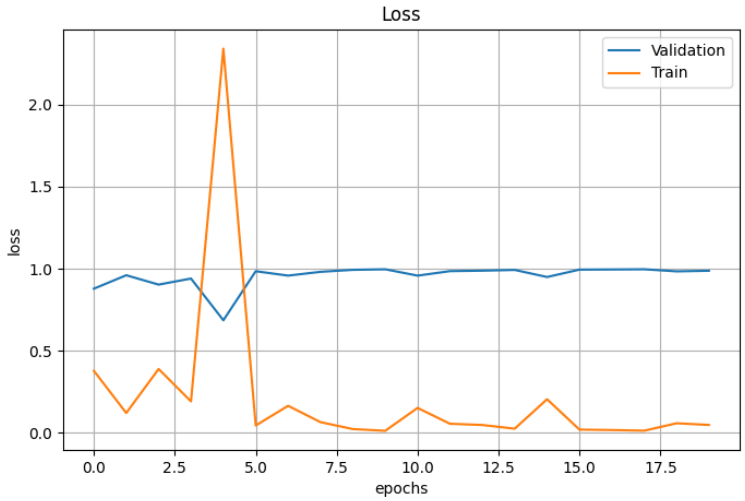

Figure A.4 Loss of E002 model while training

A.3 Training History for E003

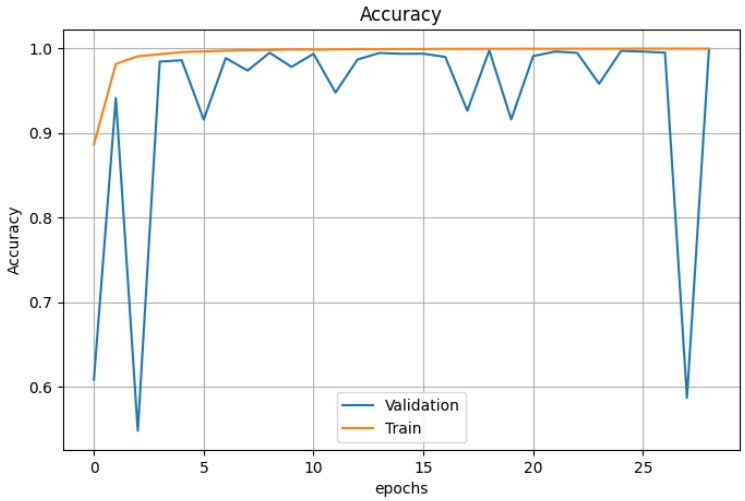

Figure A.5 Accuracy of E003 model while training

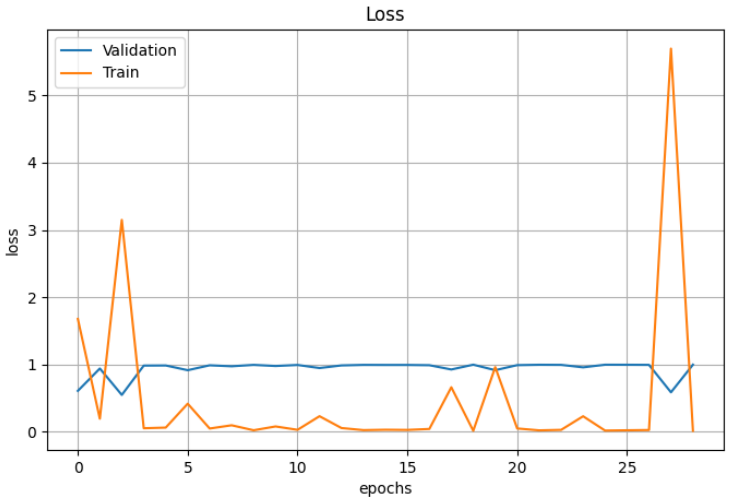

Figure A.6 Loss of E003 model while training

A.4 Training History for E004

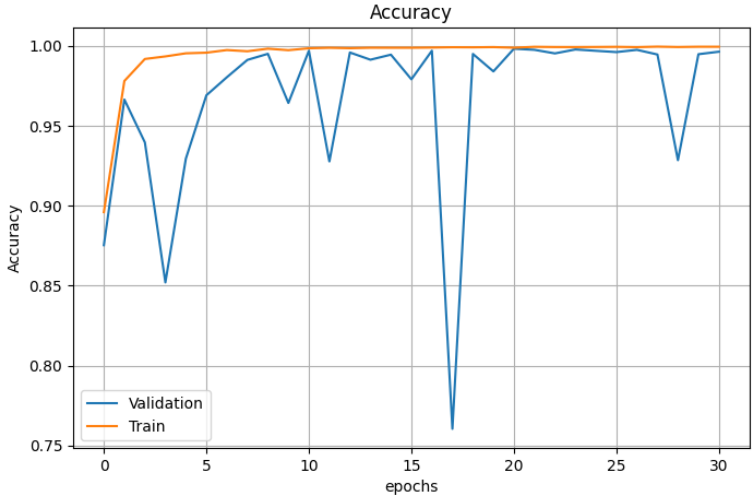

Figure A.7 Accuracy of E004 model while training

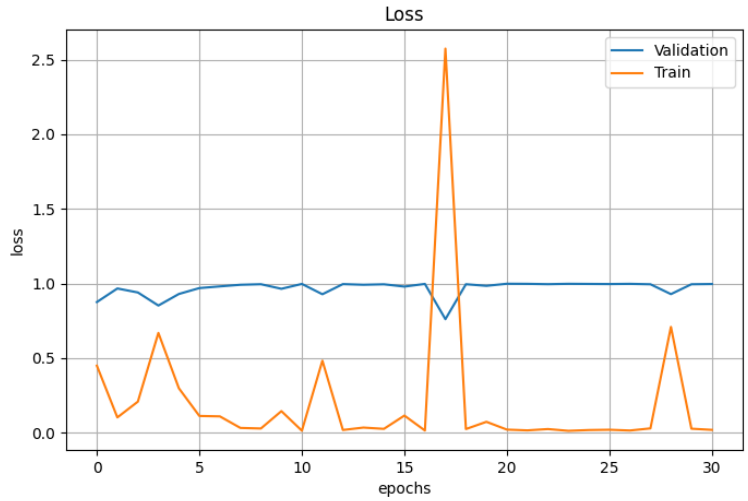

Figure A.8 Loss of E004 model while training

A.5 Training and Testing of Individual CNNs for E010

This section contains the results of train and test of the individual CNNs in the experiment E010.

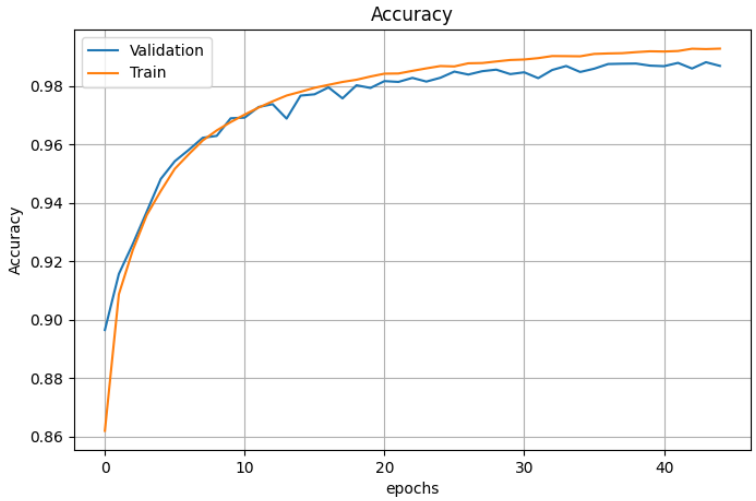

Figure A.9 Accuracy of E010/Generalist model while training

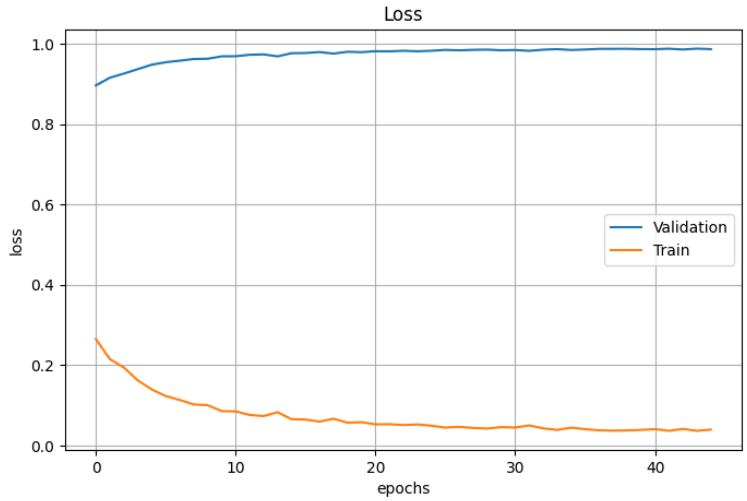

Figure A.10 Loss of E010/Generalist model while training

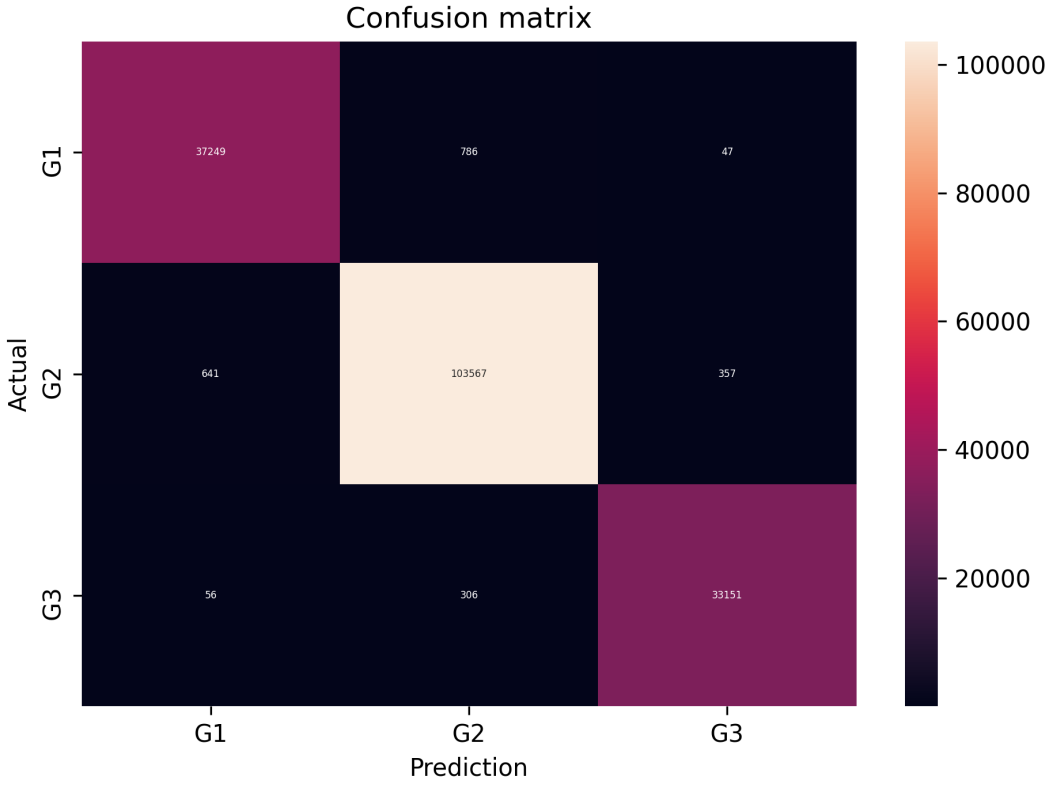

Figure A.11 Confusion matrix of E010/Generalist model

Table A.1 Evaluation metrics for E010/Generalist model on test dataset

	precision	recall	f1-score	support
G1	98.16	97.81	97.99	38082
G2	98.96	99.05	99.00	104565
G3	98.80	98.92	98.86	33513
accuracy			98.76	176160
macro avg	98.64	98.59	98.62	176160
weighted avg	98.75	98.76	98.75	176160

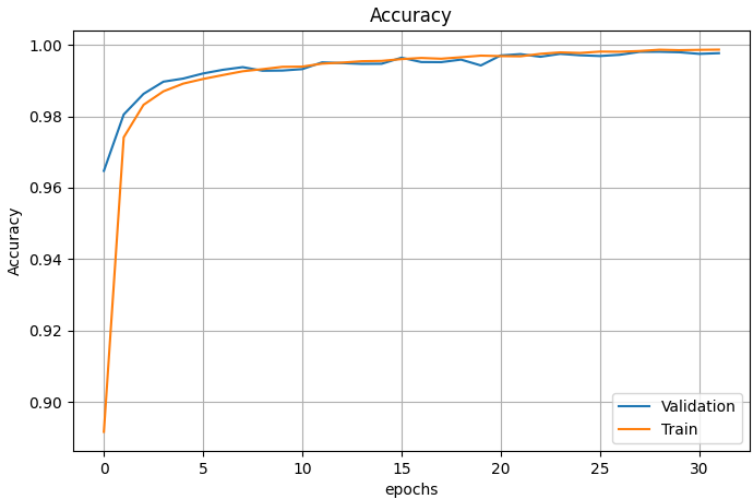

Figure A.12 Accuracy of E010/G1 model while training

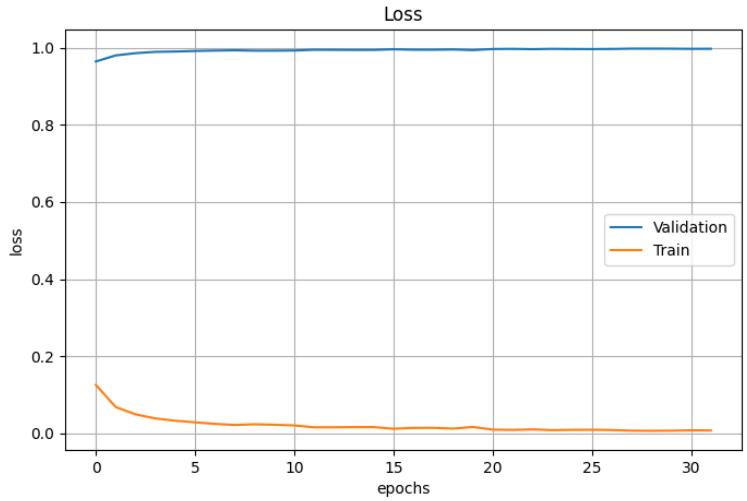

Figure A.13 Loss of E010/G1 model while training

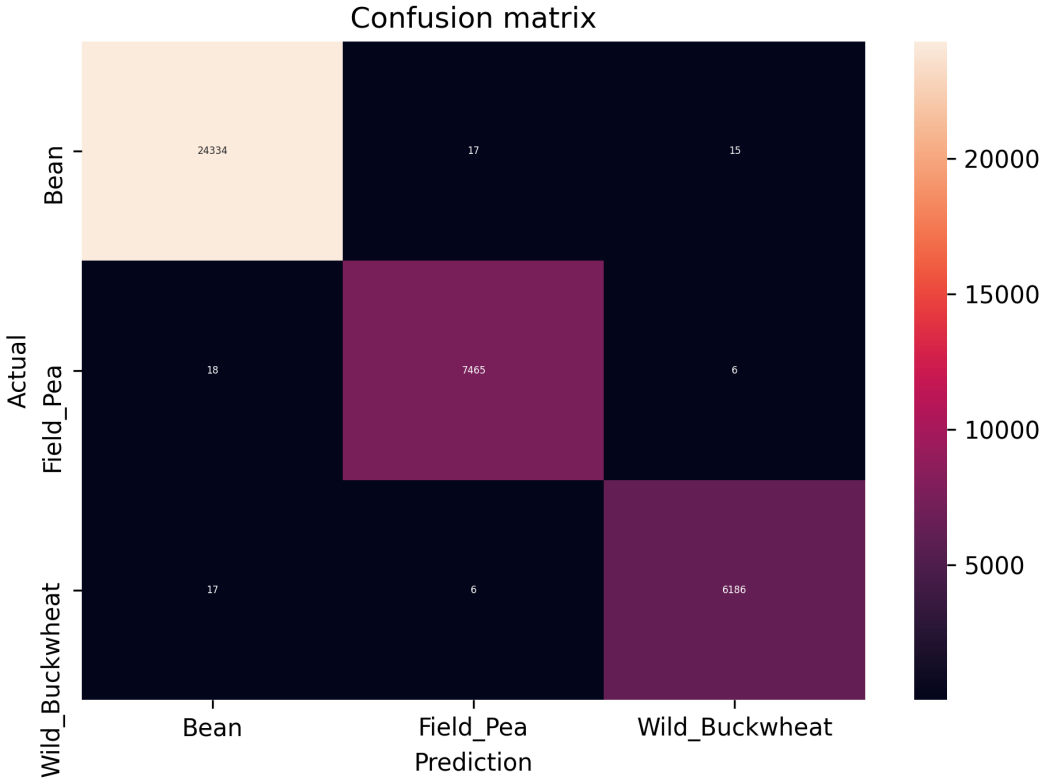

Figure A.14 Confusion matrix of E010/G1 model

Table A.2 Evaluation metrics for E010/G1 model on test dataset

	precision	recall	f1-score	support
Bean	99.86	99.87	99.86	24366
Field_Pea	99.69	99.68	99.69	7489
Wild_Buckwheat	99.66	99.63	99.65	6209
accuracy			99.79	38064
macro avg	99.74	99.73	99.73	38064
weighted avg	99.79	99.79	99.79	38064

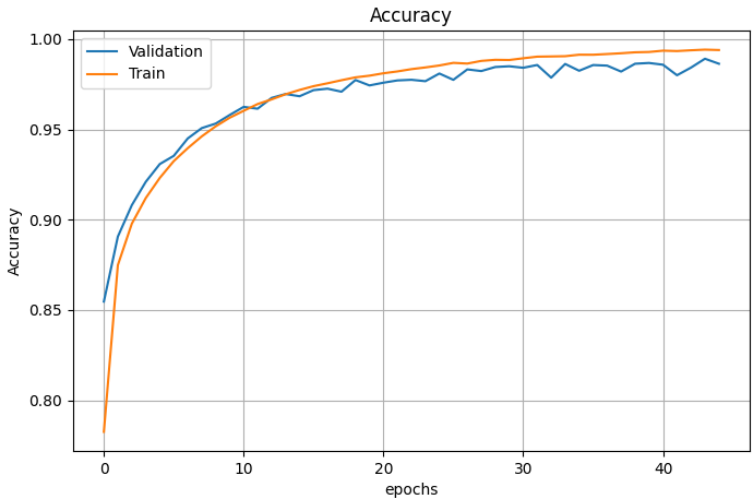

Figure A.15 Accuracy of E010/G2 model while training

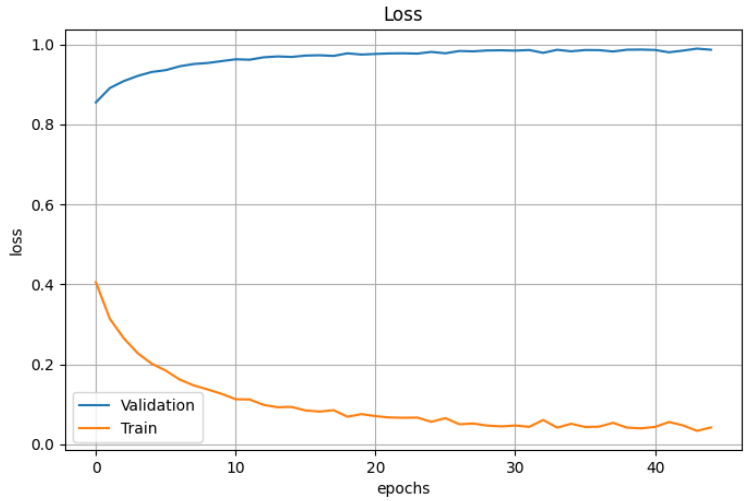

Figure A.16 Loss of E010/G2 model while training

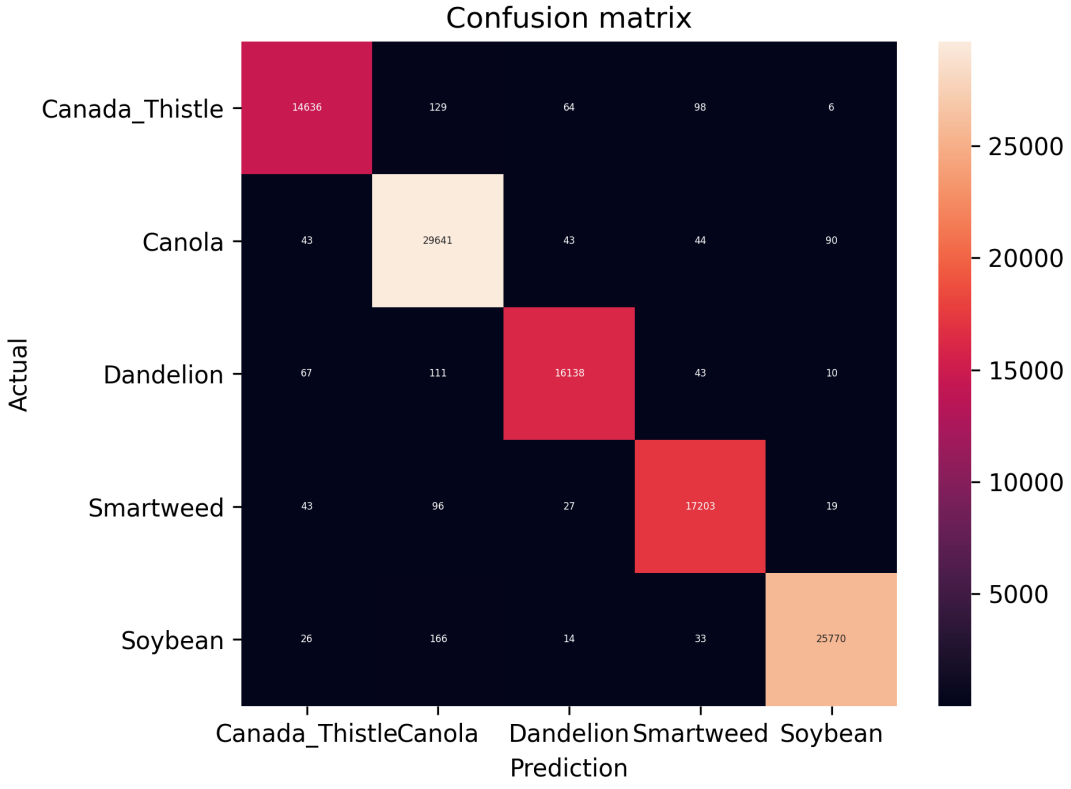

Figure A.17 Confusion matrix of E010/G2 model

Table A.3 Evaluation metrics for E010/G2 model on test dataset

	precision	recall	f1-score	support
Canada_Thistle	98.79	98.01	98.40	14933
Canola	98.33	99.26	98.80	29861
Dandelion	99.09	98.59	98.84	16369
Smartweed	98.75	98.94	98.84	17388
Soybean	99.52	99.08	99.30	26009
accuracy			98.88	104560
macro avg	98.90	98.78	98.84	104560
weighted avg	98.88	98.88	98.88	104560

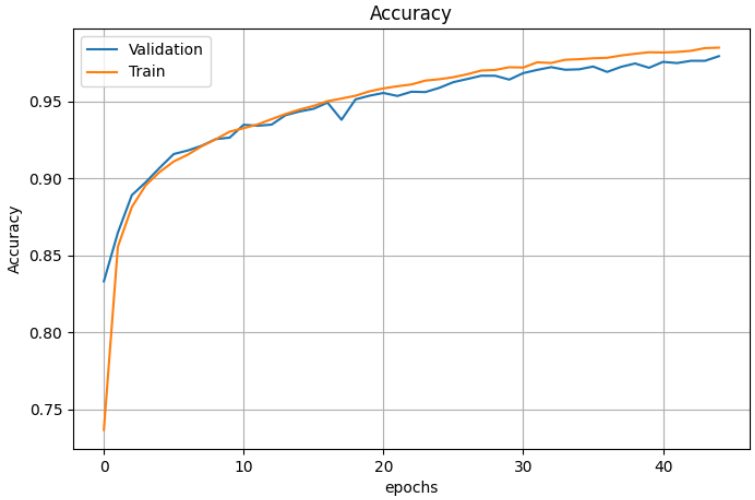

Figure A.18 Accuracy of E010/G3 model while training

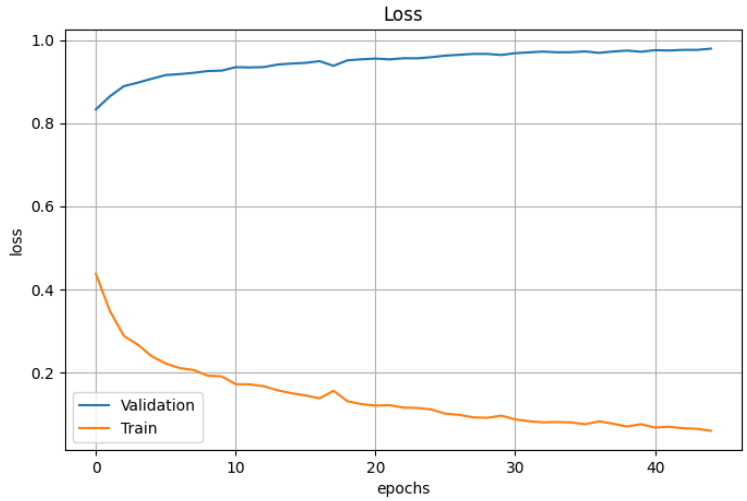

Figure A.19 Loss of E010/G3 model while training

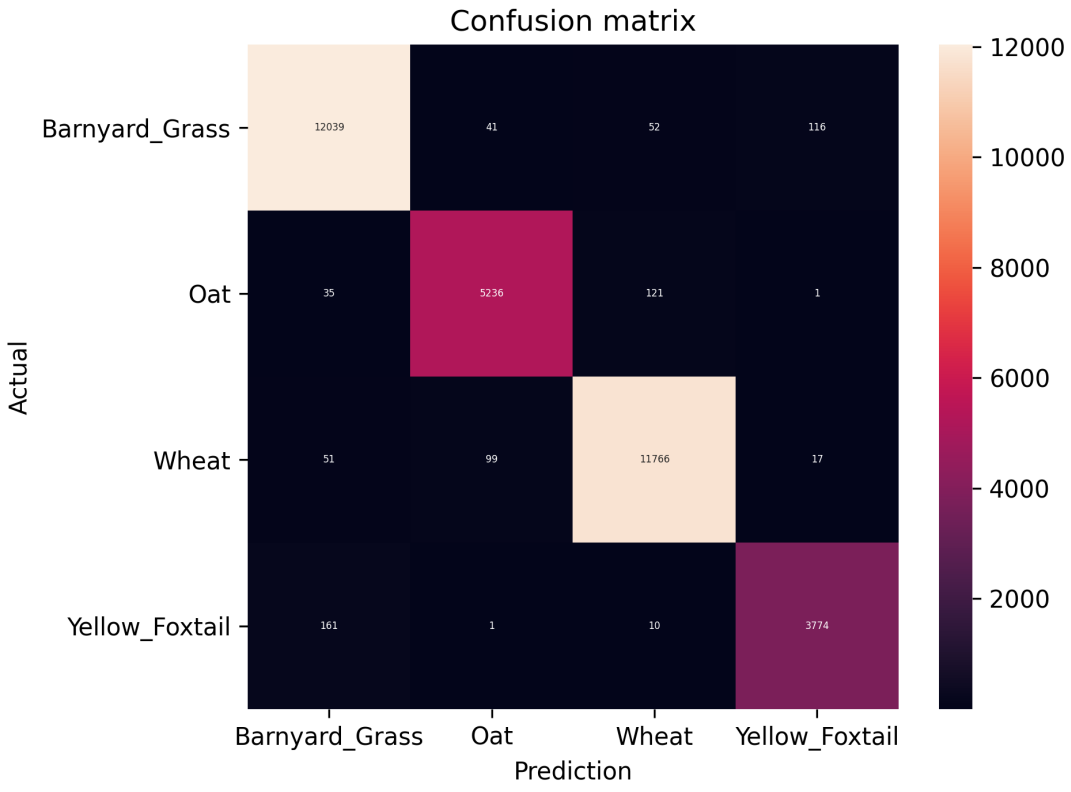

Figure A.20 Confusion matrix of E010/G3 model

Table A.4 Evaluation metrics for E010/G3 model on test dataset

	precision	recall	f1-score	support
Barnyard_Grass	97.99	98.29	98.14	12248
Oat	97.38	97.09	97.23	5393
Wheat	98.47	98.60	98.53	11933
Yellow_Foxtail	96.57	95.64	96.10	3946
accuracy			97.90	33520
macro avg	97.60	97.41	97.50	33520
weighted avg	97.89	97.90	97.90	33520

A.6 Training and Testing of Individual CNNs for E011

This section contains the results of train and test of the individual CNNs in the experiment E011.

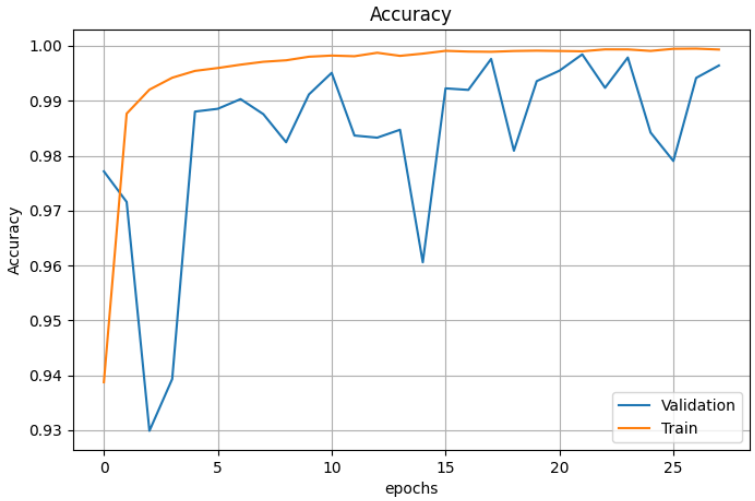

Figure A.21 Accuracy of E011/Generalist model while training

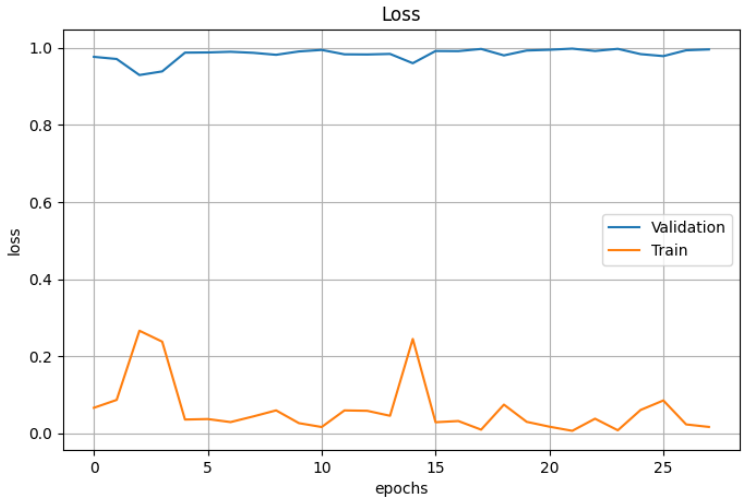

Figure A.22 Loss of E011/Generalist model while training

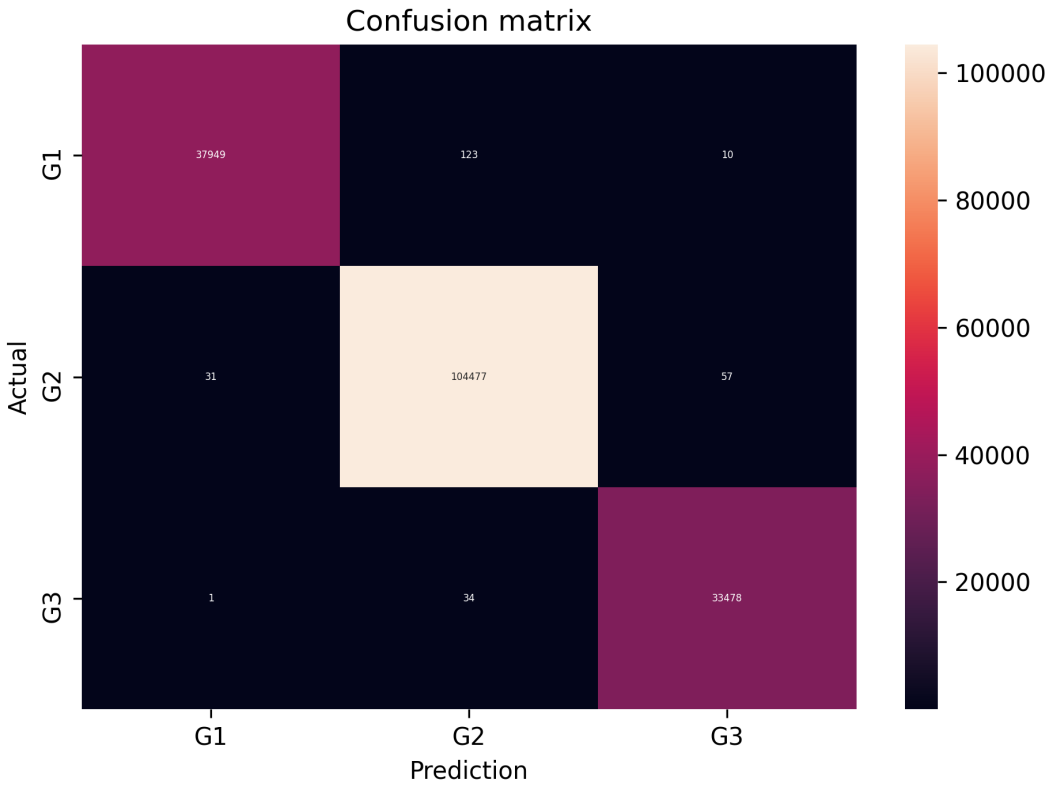

Figure A.23 Confusion matrix of E011/Generalist model

Table A.5 Evaluation metrics for E011/Generalist model on test dataset

	precision	recall	f1-score	support
G1	99.92	99.65	99.78	38082
G2	99.85	99.92	99.88	104565
G3	99.80	99.90	99.85	33513
accuracy			99.85	176160
macro avg	99.86	99.82	99.84	176160
weighted avg	99.85	99.85	99.85	176160

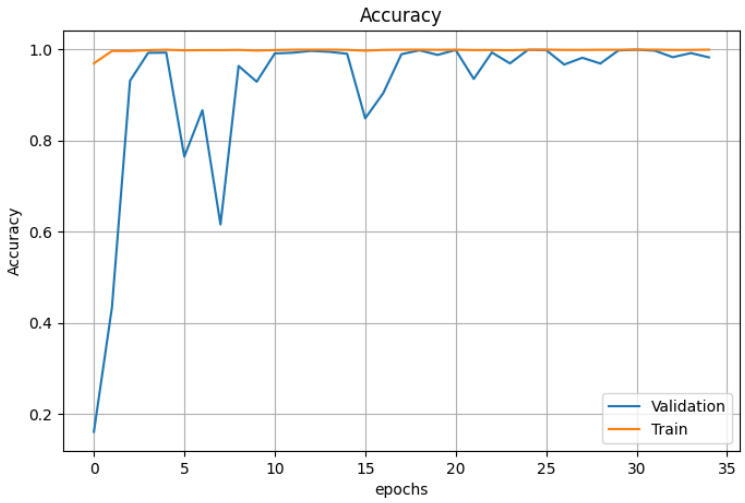

Figure A.24 Accuracy of E011/G1 model while training

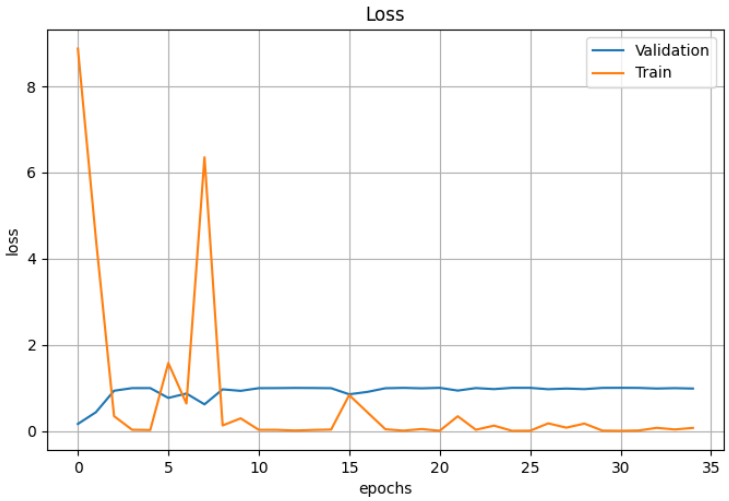

Figure A.25 Loss of E011/G1 model while training

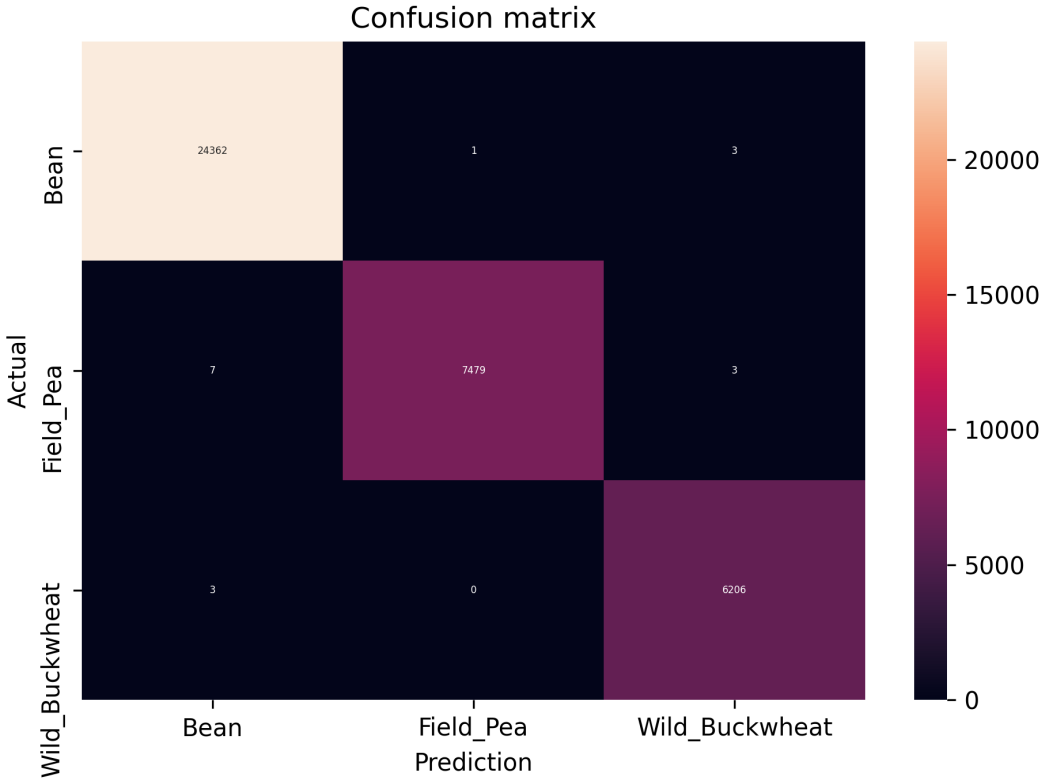

Figure A.26 Confusion matrix of E011/G1 model

Table A.6 Evaluation metrics for E011/G1 model on test dataset

	precision	recall	f1-score	support
Bean	99.96	99.98	99.97	24366
Field_Pea	99.99	99.87	99.93	7489
Wild_Buckwheat	99.90	99.95	99.93	6209
accuracy			99.96	38064
macro avg	99.95	99.93	99.94	38064
weighted avg	99.96	99.96	99.96	38064

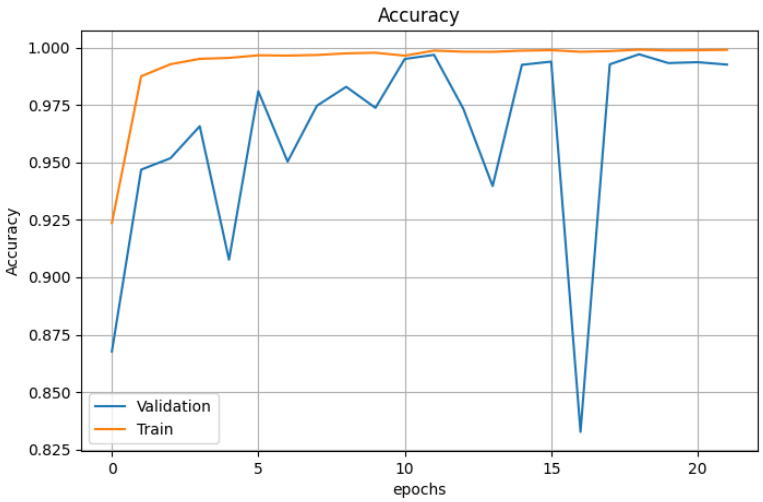

Figure A.27 Accuracy of E011/G2 model while training

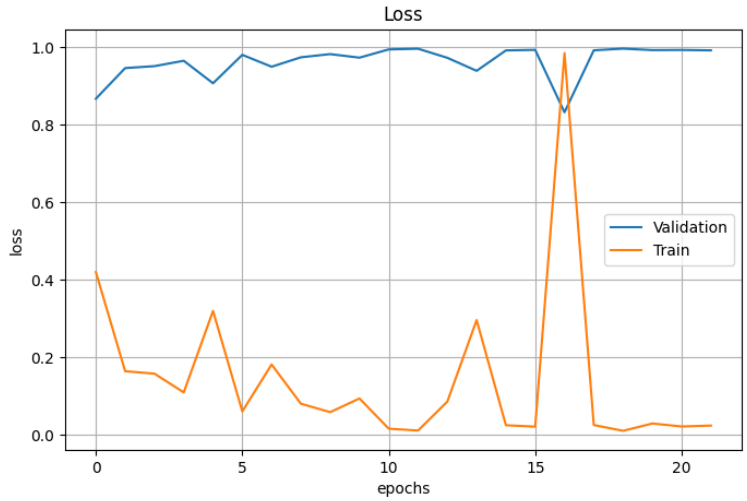

Figure A.28 Loss of E011/G2 model while training

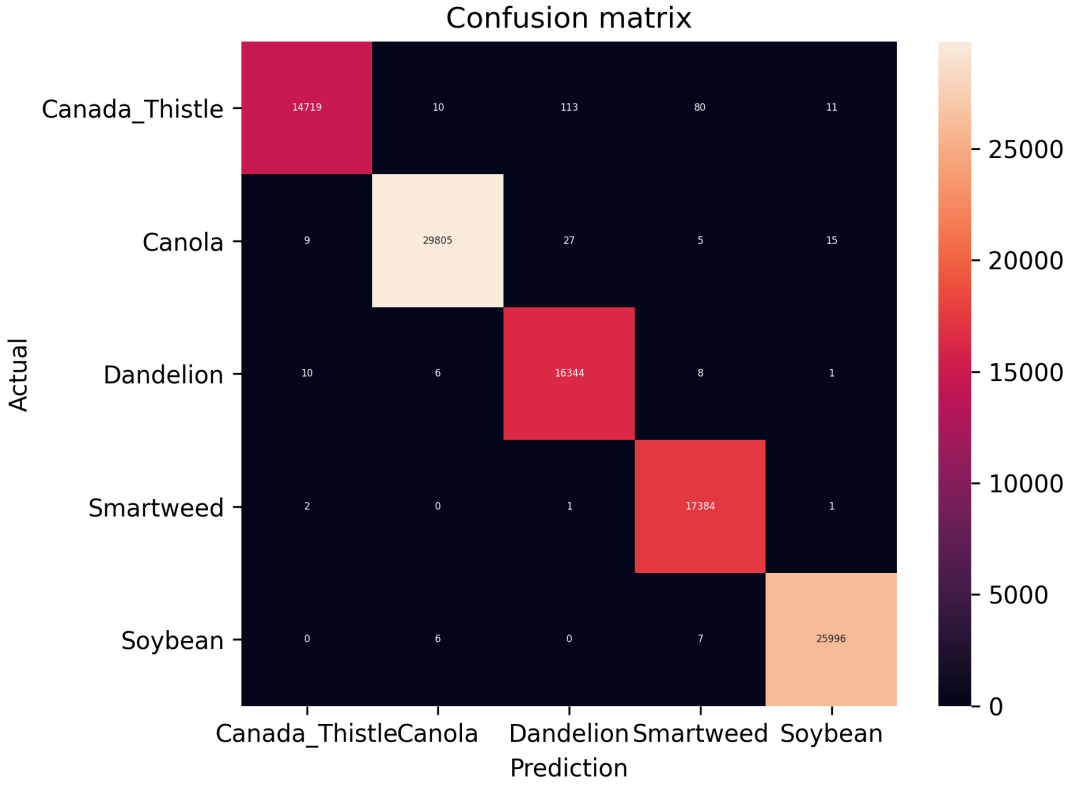

Figure A.29 Confusion matrix of E011/G2 model

Table A.7 Evaluation metrics for E011/G2 model on test dataset

	precision	recall	f1-score	support
Canada_Thistle	99.86	98.57	0.9921	14933
Canola	99.93	99.81	0.9987	29861
Dandelion	99.14	99.85	0.9949	16369
Smartweed	99.43	99.98	0.9970	17388
Soybean	99.89	99.95	0.9992	26009
accuracy			0.9970	104560
macro avg	99.65	99.63	0.9964	104560
weighted avg	99.70	99.70	0.9970	104560

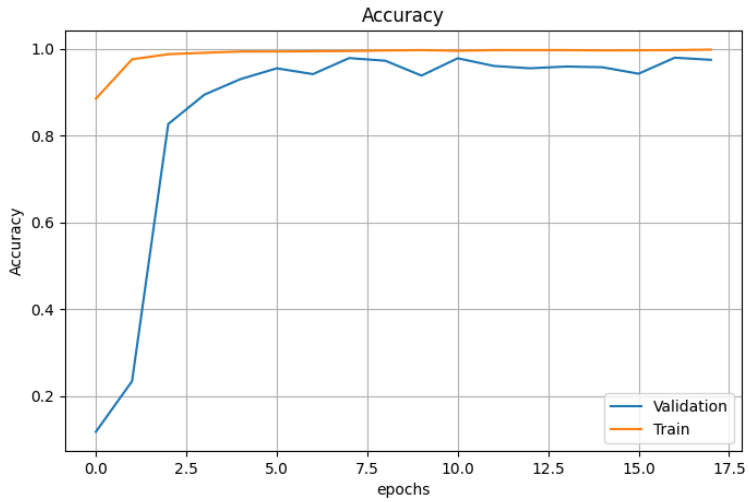

Figure A.30 Accuracy of E011/G3 model while training

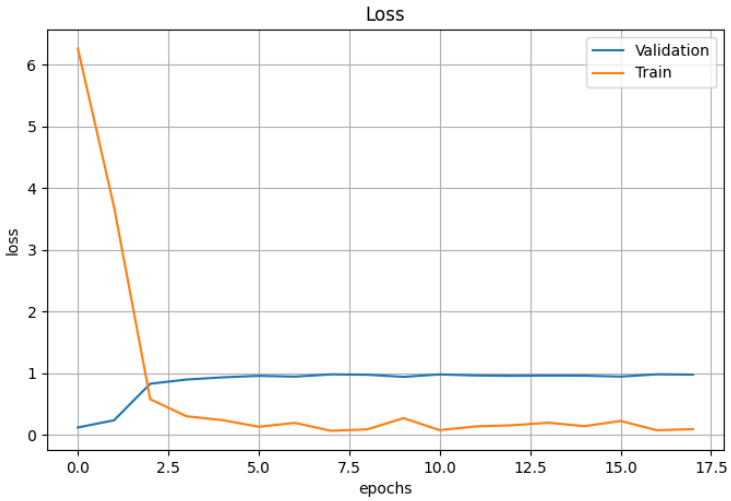

Figure A.31 Loss of E011/G3 model while training

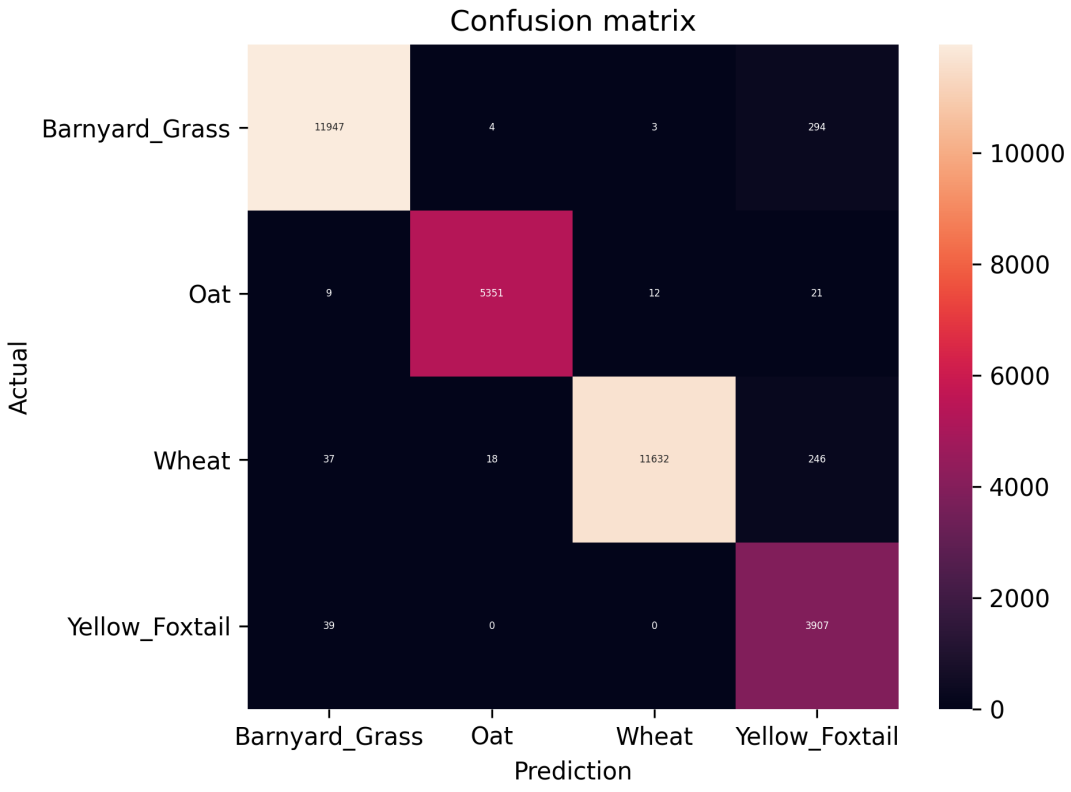

Figure A.32 Confusion matrix of E011/G3 model

Table A.8 Evaluation metrics for E011/G3 model on test dataset

	precision	recall	f1-score	support
Barnyard_Grass	99.29	97.54	98.41	12248
Oat	99.59	99.22	99.41	5393
Wheat	99.87	97.48	98.66	11933
Yellow_Foxtail	87.44	99.01	92.87	3946
accuracy			97.96	33520
macro avg	96.55	98.31	97.34	33520
weighted avg	98.15	97.96	98.01	33520

A.7 Testing of Individual CNNs for E012

This section contains the results of train and test of the individual CNNs in the experiment E012.

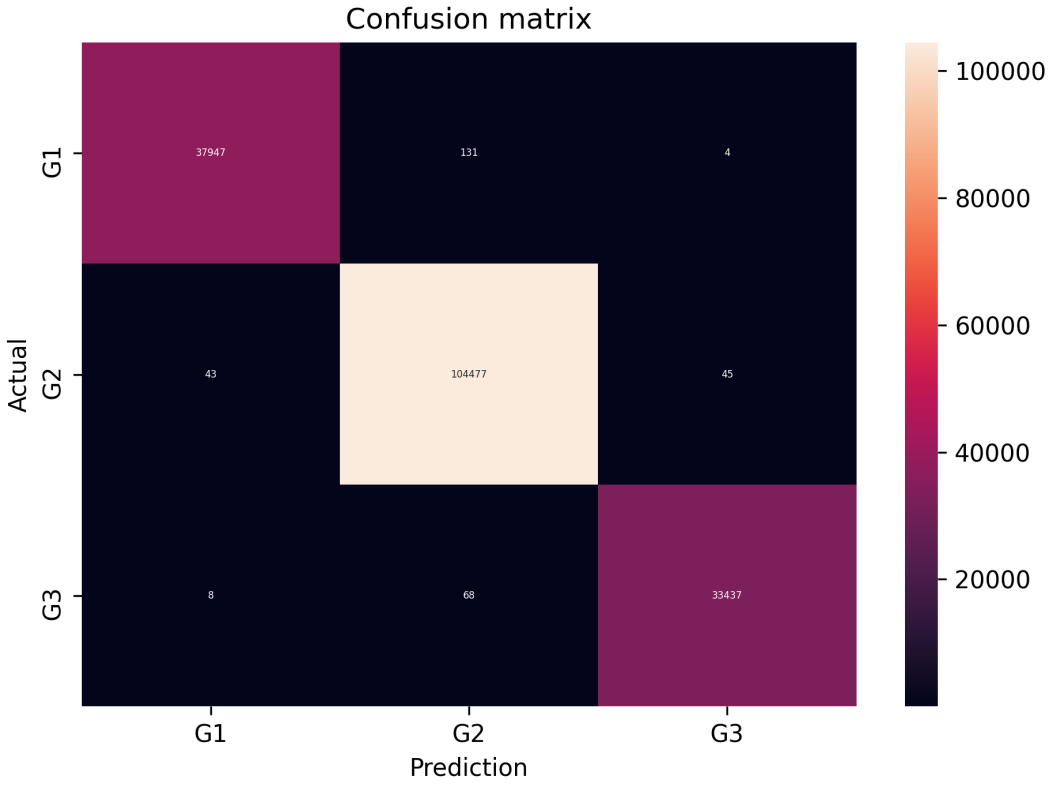

Figure A.33 Confusion matrix of E012/Generalist model

Table A.9 Evaluation metrics for E012/Generalist model on test dataset

	precision	recall	f1-score	support
G1	99.87	99.65	99.76	38082
G2	99.81	99.92	99.86	104565
G3	99.85	99.77	99.81	33513
accuracy			99.83	176160
macro avg	99.84	99.78	99.81	176160
weighted avg	99.83	99.83	99.83	176160

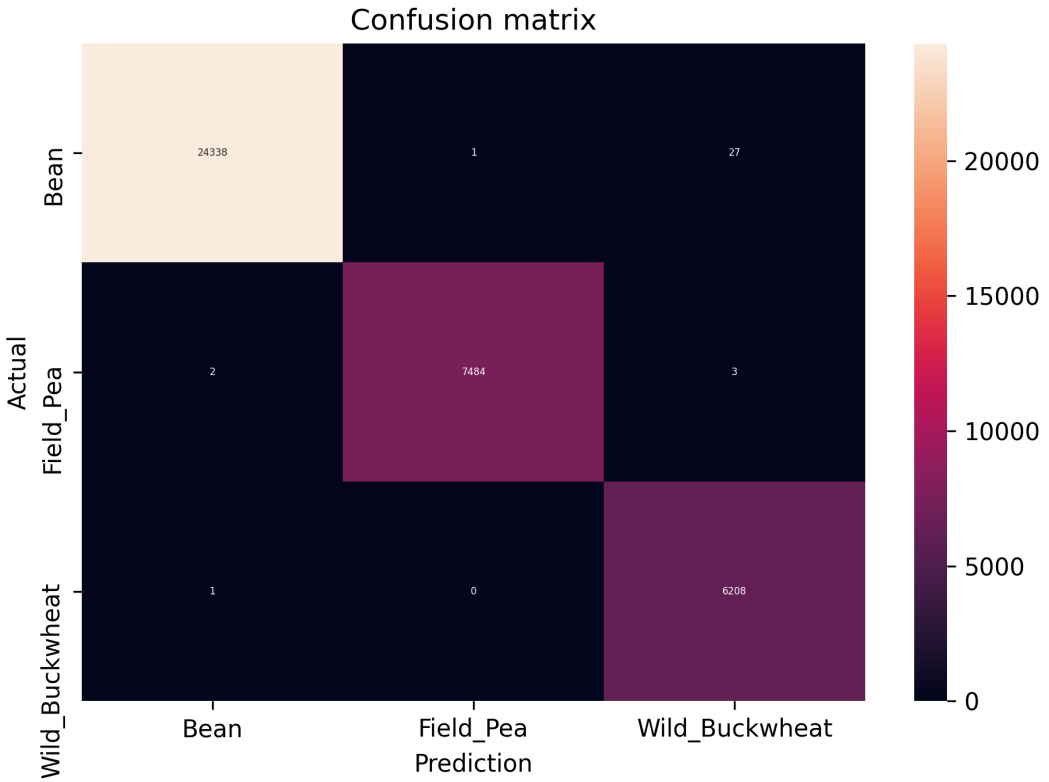

Figure A.34 Confusion matrix of E012/G1 model

Table A.10 Evaluation metrics for E012/G1 model on test dataset

	precision	recall	f1-score	support
Bean	99.99	99.89	99.94	24366
Field_Pea	99.99	99.93	99.96	7489
Wild_Buckwheat	99.52	99.98	99.75	6209
accuracy			99.91	38064
macro avg	99.83	99.93	99.88	38064
weighted avg	99.91	99.91	99.91	38064

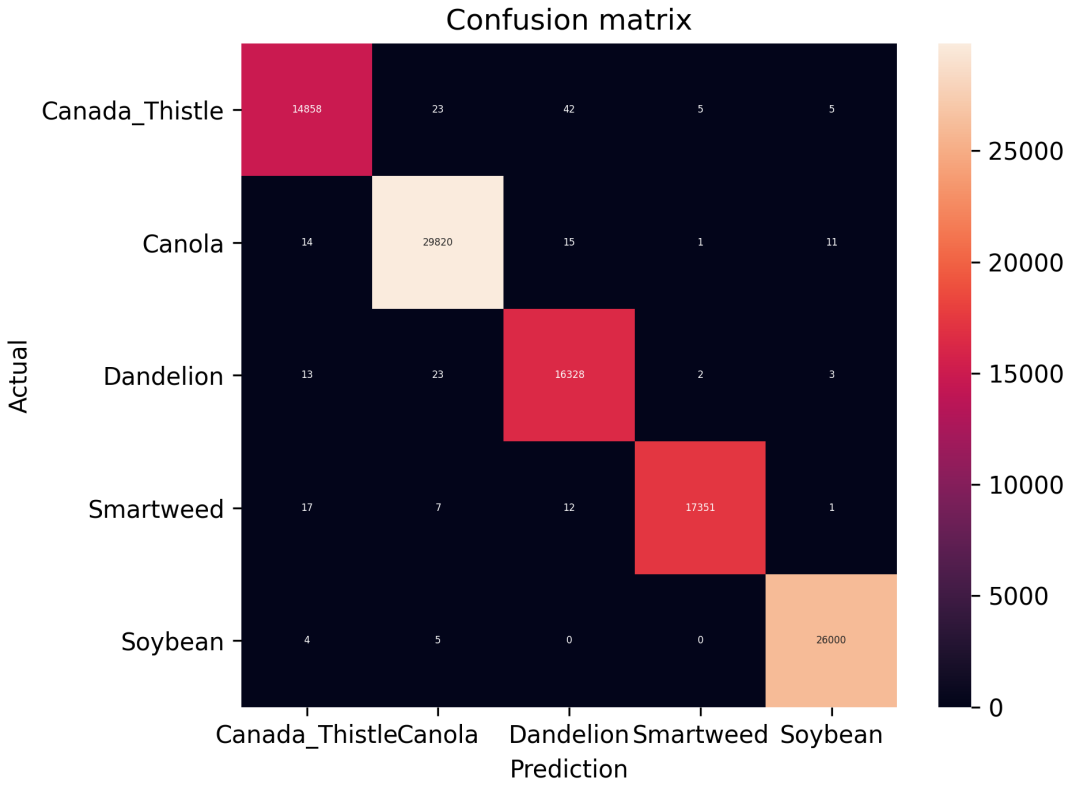

Figure A.35 Confusion matrix of E012/G2 model

Table A.11 Evaluation metrics for E012/G2 model on test dataset

	precision	recall	f1-score	support
Canada_Thistle	99.68	99.50	99.59	14933
Canola	99.81	99.86	99.83	29861
Dandelion	99.58	99.75	99.66	16369
Smartweed	99.95	99.79	99.87	17388
Soybean	99.92	99.97	99.94	26009
accuracy			99.81	104560
macro avg	99.79	99.77	99.78	104560
weighted avg	99.81	99.81	99.81	104560

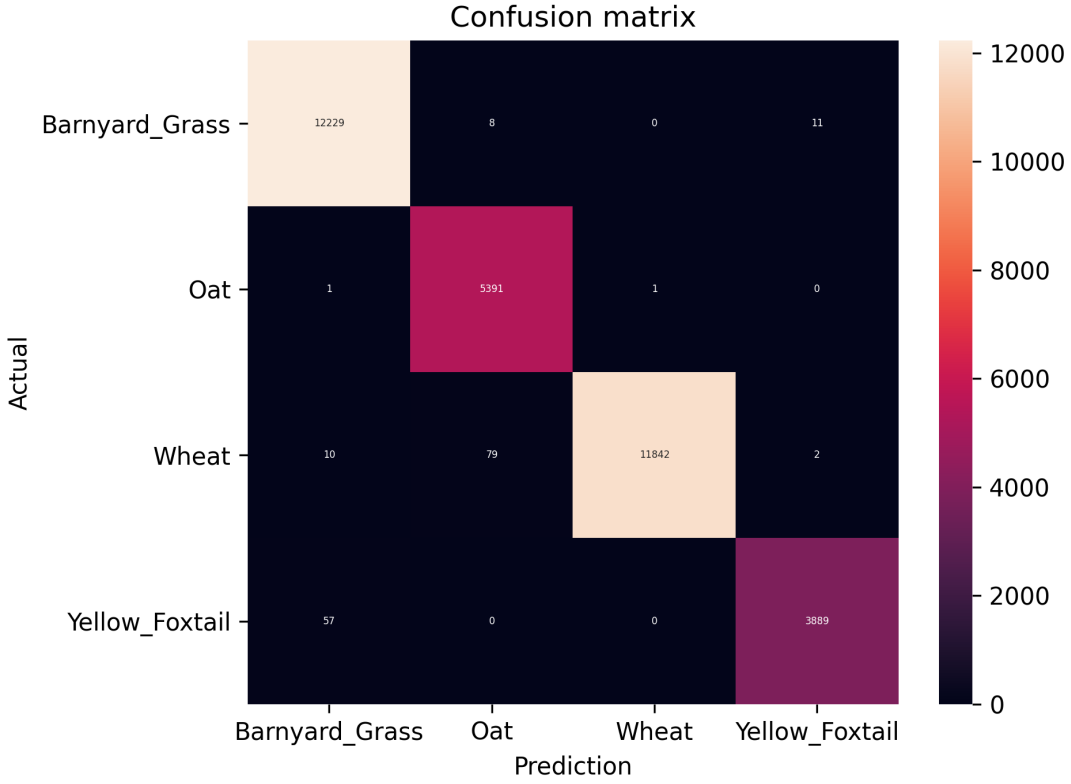

Figure A.36 Confusion matrix of E012/G3 model

Table A.12 Evaluation metrics for E012/G3 model on test dataset

	precision	recall	f1-score	support
Barnyard_Grass	99.45	99.84	99.65	12248
Oat	98.41	99.96	99.18	5393
Wheat	99.99	99.24	99.61	11933
Yellow_Foxtail	99.67	98.56	99.11	3946
accuracy			99.50	33520
macro avg	99.38	99.40	99.39	33520
weighted avg	99.50	99.50	99.50	33520

A.8 Test Results for E001 on Weed Seedling Dataset

Table A.13 Evaluation metrics for E001 model on test dataset, SameAngles

	precision	recall	f1-score	support
Barnyard Grass	84.38	07.71	14.14	350
Bean	00.00	00.00	00.00	0
Canada Thistle	47.72	18.65	26.82	504
Canola	33.43	48.90	39.72	456
Dandelion	90.23	54.34	67.83	357
Field Pea	00.00	00.00	00.00	0
Oat	00.00	00.00	00.00	0
Smartweed	35.29	94.44	51.38	504
Soybean	00.00	00.00	00.00	0
Wheat	00.00	00.00	00.00	0
Wild Buckwheat	87.06	68.23	76.50	513
Yellow Foxtail	94.39	45.68	61.56	810
accuracy			49.63	3494
macro avg	39.37	28.16	28.16	3494
weighted avg	68.67	49.63	50.31	3494

Table A.14 Evaluation metrics for E001 model on test dataset, RandomAngles

	precision	recall	f1-score	support
Barnyard Grass	71.43	04.42	08.33	113
Bean	00.00	00.00	00.00	0
Canada Thistle	80.49	60.00	68.75	55
Canola	54.14	75.22	62.96	113
Dandelion	97.37	61.67	75.51	60
Field Pea	00.00	00.00	00.00	0
Smartweed	28.42	88.52	43.03	61
Soybean	00.00	00.00	00.00	0
Wheat	00.00	00.00	00.00	0
Wild Buckwheat	80.00	07.02	12.90	57
Yellow Foxtail	48.28	22.95	31.11	61
accuracy			44.62	520
macro avg	41.83	29.07	27.51	520
weighted avg	64.80	44.62	41.59	520

Table A.15 Evaluation metrics for E001 model on test dataset, Smartphone

	precision	recall	f1-score	support
Barnyard Grass	00.00	00.00	00.00	8
Bean	00.00	00.00	00.00	0
Canada Thistle	14.29	12.50	13.33	8
Canola	23.08	37.50	28.57	8
Dandelion	22.22	25.00	23.53	8
Field Pea	00.00	00.00	00.00	0
Oat	00.00	00.00	00.00	0
Smartweed	16.67	12.50	14.29	8
Soybean	00.00	00.00	00.00	0
Wheat	00.00	00.00	00.00	0
Wild Buckwheat	00.00	00.00	00.00	8
Yellow Foxtail	00.00	00.00	00.00	8
accuracy			12.50	56
macro avg	06.35	07.29	06.64	56
weighted avg	10.89	12.50	11.39	56

A.9 Test Results for E002 on Weed Seedling Dataset

Table A.16 Evaluation metrics for E002 model on test dataset, SameAngles

	precision	recall	f1-score	support
Barnyard Grass	28.57	00.57	01.12	350
Bean	00.00	00.00	00.00	0
Canada Thistle	62.02	60.91	61.46	504
Canola	46.67	92.32	62.00	456
Dandelion	53.60	18.77	27.80	357
Field Pea	00.00	00.00	00.00	0
Oat	00.00	00.00	00.00	0
Smartweed	28.80	39.48	33.31	504
Soybean	00.00	00.00	00.00	0
Wheat	00.00	00.00	00.00	0
Wild Buckwheat	76.66	74.27	75.45	513
Yellow Foxtail	91.52	63.95	75.29	810
accuracy			54.24	3494
macro avg	32.32	29.19	28.04	3494
weighted avg	60.00	54.24	53.25	3494

Table A.17 Evaluation metrics for E002 model on test dataset, RandomAngles

	precision	recall	f1-score	support
Barnyard Grass	33.33	00.88	01.72	113
Bean	00.00	00.00	00.00	0
Canada Thistle	55.56	72.73	62.99	55
Canola	56.02	82.30	66.67	113
Dandelion	29.09	26.67	27.83	60
Field Pea	00.00	00.00	00.00	0
Oat	00.00	00.00	00.00	0
Smartweed	46.51	65.57	54.42	61
Soybean	00.00	00.00	00.00	0
Wheat	00.00	00.00	00.00	0
Wild Buckwheat	54.05	35.09	42.55	57
Yellow Foxtail	40.82	32.79	36.36	61
accuracy			44.23	520
macro avg	26.28	26.34	24.38	520
weighted avg	44.82	44.23	40.05	520

Table A.18 Evaluation metrics for E002 model on test dataset, Smartphone

	precision	recall	f1-score	support
Barnyard Grass	100.00	12.50	22.22	8
Bean	00.00	00.00	00.00	0
Canada Thistle	16.67	12.50	14.29	8
Canola	14.29	25.00	18.18	8
Dandelion	22.22	25.00	23.53	8
Field Pea	00.00	00.00	00.00	0
Smartweed	33.33	12.50	18.18	8
Soybean	00.00	00.00	00.00	0
Wheat	00.00	00.00	00.00	0
Wild Buckwheat	100.00	100.00	100.00	8
Yellow Foxtail	16.67	12.50	14.29	8
accuracy			14.29	56
macro avg	18.47	09.09	10.06	56
weighted avg	29.02	14.29	15.81	56

A.10 Training and Testing for E001 on Sightline data

Followings are the tests that are summariesed in Table 5.9 of Section 5.3.

Table A.19 Evaluation metrics for e001-sl model on test dataset, SameAngles

	precision	recall	f1-score	support
Barnyard Grass	96.15	07.14	13.30	350
Canada Thistle	60.54	26.79	37.14	504
Canola	43.09	47.15	45.03	456
Dandelion	87.22	55.46	67.81	357
Smartweed	32.03	96.03	48.04	504
Wild Buckwheat	89.80	68.62	77.79	513
Yellow Foxtail	81.82	62.22	70.69	810
accuracy			54.75	3494
macro avg	70.09	51.92	51.40	3494
weighted avg	69.67	54.75	54.23	3494

Table A.20 Evaluation metrics for e001-sl model on test dataset, RandomAngles

	precision	recall	f1-score	support
Barnyard Grass	77.78	12.39	21.37	113
Canada Thistle	83.78	56.36	67.39	55
Canola	64.33	89.38	74.81	113
Dandelion	83.33	41.67	55.56	60
Smartweed	25.12	83.61	38.64	61
Wild Buckwheat	40.00	03.51	06.45	57
Yellow Foxtail	60.00	68.85	64.12	61
accuracy			51.15	520
macro avg	62.05	50.82	46.91	520
weighted avg	63.73	51.15	47.20	520

Table A.21 Evaluation metrics for e001-sl model on test dataset, Smartphone

	precision	recall	f1-score	support
Barnyard Grass	00.00	00.00	00.00	8
Canada Thistle	50.00	25.00	33.33	8
Canola	25.00	75.00	37.50	8
Dandelion	12.50	25.00	16.67	8
Smartweed	30.00	37.50	33.33	8
Wild Buckwheat	00.00	00.00	00.00	8
Yellow Foxtail	00.00	00.00	00.00	8
accuracy			23.21	56
macro avg	16.79	23.21	17.26	56
weighted avg	16.79	23.21	17.26	56

A.11 Training and Testing for E002 on Sightline data

Followings are the tests that are summariesed in Table 5.9 of Section 5.3.

Table A.22 Evaluation metrics for e002-sl model on test dataset, SameAngles

	precision	recall	f1-score	support
Barnyard Grass	00.00	00.00	00.00	350
Canada Thistle	36.36	43.65	39.68	504
Canola	68.96	50.66	58.41	456
Dandelion	51.54	18.77	27.52	357
Smartweed	24.88	74.40	37.29	504
Wild Buckwheat	93.57	70.96	80.71	513
Yellow Foxtail	87.72	54.69	67.38	810
accuracy			48.65	3494
macro avg	51.86	44.73	44.43	3494
weighted avg	57.18	48.65	49.01	3494

Table A.23 Evaluation metrics for e002-sl model on test dataset, RandomAngles

	precision	recall	f1-score	support
Barnyard Grass	77.78	12.39	21.37	113
Canada Thistle	83.78	56.36	67.39	55
Canola	64.33	89.38	74.81	113
Dandelion	83.33	41.67	55.56	60
Smartweed	25.12	83.61	38.64	61
Wild Buckwheat	40.00	03.51	06.45	57
Yellow Foxtail	60.00	68.85	64.12	61
accuracy			51.15	520
macro avg	62.05	50.82	46.91	520
weighted avg	63.73	51.15	47.20	520

Table A.24 Evaluation metrics for e002-sl model on test dataset, Smartphone

	precision	recall	f1-score	support
Barnyard Grass	33.33	00.88	01.72	113
Canada Thistle	34.57	50.91	41.18	55
Canola	69.09	67.26	68.16	113
Dandelion	34.78	26.67	30.19	60
Smartweed	25.71	73.77	38.14	61
Wild Buckwheat	50.00	22.81	31.33	57
Yellow Foxtail	41.77	54.10	47.14	61
accuracy			40.77	520
macro avg	41.32	42.34	36.84	520
weighted avg	43.32	40.77	36.46	520