2021

Plant Species Identification In The Wild Based On Images Of Organs

Meghana Kovur
West Virginia University, mk0174@mix.wvu.edu

Plant Species Identification In The Wild Based On Images Of Organs

Meghana Kovur

Problem Report submitted to the
Statler College of Engineering and Mineral Resources
at West Virginia University

In partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Gianfranco Doretto, PhD., Chair
Donald Adjeroh, Ph.D.
Katerina Goseva-Popstojanova, Ph.D.

LANE DEPARTMENT OF COMPUTER SCIENCE
AND ELECTRICAL ENGINEERING

West Virginia University

Morgantown, WV
July 30, 2021

Keywords: Plant Species Classification, FGVC, Classification, Fine-Grained Visual Classification, Optimization

Abstract

Plant Species Identification In The Wild Based On Images Of Organs

by Meghana Kovur

Image-based plant species identification in the wild is a difficult problem for several reasons. First, the input data is subject to a very high degree of variability because it is captured under fully unconstrained conditions. The same plant species may look very different in different images, while different species can often appear very similar, challenging even the recognition skills of human experts in the field. The large intra-class and small inter-class image variability makes this a fine-grained visual classification problem. One way to cope with this variability and to reduce image background noise is to predict species based on the plant organs that are visible. In this work, we designed a set of classifiers that predict species conditioned on the plant organs. Due to the lack of a dataset for testing this task, we curated one ourselves with 1000 species and 5 organ types per species, which is based on the data used in the PlantCLEF 2015 challenge. We designed the bank of organ-based plant species classifiers with convolutional neural networks, and we performed a comparative study that shows how different techniques for improving the classifier training affect the accuracy of the identification for each of the organ type.

Acknowledgements

Most importantly I would like to thank Dr. Gianfranco Doretto for giving me an opportunity to be part of his Vision Lab. It is because of his motivation to accomplish things, constant follow-ups and engaging with the group which helped us achieve our goals. Without him, I would not have learnt how one contributes to a research. Thank you for teaching me and handling me through the process. I would like to thank Matthew Keaton, Ram Jitendrabhai Zaveri for being the best team players. It is because of their continuous contribution to this project, we were able to successfully accomplish many tasks. Without them, it is almost impossible to publish a paper. I would like to thank Zaigham Randhawa, Ranya Almohsen, Shivang Patel and Quinn Jones for always keeping the group alive and for being a backbone to the lab work. All my initial accomplishments would not have been possible without them.

I would like to thank Dr. Brain Powell for giving me an opportunity to work with him in the Computer Science Department. I would sincerely thank Akhil Bharthavarapu for guiding me through most important aspects of the Masters Program. Without his inital guidance it would not have been possible for me to be a part of the WVU. I would appreciate Sai Pushpitha Vudatha, Surekha Pachipulusu for accepting me into their lives and nurturing me, without them it would have taken ages to familiarize myself with Morgantown. I would thank Savan Suri, Sujan Kasani and Ranaa Mahveen for motivating and guiding me when I needed the most. My special thanks to Sasanka Katreddi for always being with me, helping and guiding me. Without her I would not have felt home away from home. I thank Niharika Alahari, Ashish Maradapa for constant check on me. I thank Prathyusha Vakkantula, Akash Mehta, Sahithi, Nikitha, Harika, Sharath, Pavan, Chakri, Vinay, Raghu Vamsi for making my life easy in Morgantown. I thank Rohith and Harshini for being my good course mates, they made sure that I don't feel left out. My sincere thanks to Sai Satish Guda, Vivek Kommarina, Hemanth, Nagarjuna Elisetty, Anudeep for making me feel at home and being around.

For me, this Masters Program would not have been possible without constant moral support and love from my parents. I cannot say how thankful I am to my sister Sneha Varsha, for always hearing my tantrums. Without her constant support I cannot imagine my life. My sincere thanks to Aravind Cheruvu, Vinay Cheguri, Krishna Celupuri, Arjun for being my backbone and keeping my spirits high. I thank all my friends back home, without whom I would not have been as lively as I am today.

Contents

List of Figures

List of Tables

List of Abbreviations

FGVC	**F**ine **G**rained **V**isual **C**lassification
DARMA	**D**etection **A**s **R**einforced **M**eans of **A**ttention
LR	**L**earning **R**ate
LS	**L**abel **S**moothing
ROI	**R**egion **O**f **I**nterest
CNN	**C**onvolutional **N**eural **N**etwork
BL	**B**ase**L**ine
KD	**K**nowledge **D**istillation

Chapter 1

Introduction

1.1 Background

Traditional computer vision research concentrated on the coarse-grained identification and classification of objects on a large-scale perspective. With the development of computer vision and deep learning, Fine Grained Visual Classification (FGVC) and retrieval in the wild became topical as it aims to recognize images belonging to multiple sub-categories within a same category. Plant classification and recognition have several applications in computer vision such as field guides to automate the identification of plants, registering and mapping the plants from different species for biodiversity monitoring as well as identifying and treating diseases.

Identification of plant organ species in the wild is a Fine-Grained Visual Classification problem, as there is large intra-class and small inter-class variability in the organ dataset. Due to the fact that there is a high intra class and low inter class variability in FGVC datasets, collecting fine grained visuals "in the wild" is a difficult problem compared to the conventional problem as collecting and classifying the data requires high-level domain specific knowledge. Data gathered by the citizen scientists in the wild is recognized as an effective tool for developing the knowledge and expanding the research opportunities. Although there are FGVC datasets which are used to build models for image detection or classification tasks, the data required to perform Fine-Grained Visual Classification is insufficient, most importantly "in the wild".

1.2 Motivation

The image data that is acquired in the wild effect multiple factors including illumination of the image, background, occlusion, size, shape and scale of the plant images is unconstrained. Irrespective of the variance, these different kinds of data pose additional challenges in contrast with conventional images. The recent approach to FGVC cope with that variability by leveraging attention based mechanism [1] where they focus on reducing the sources of nuisance factors

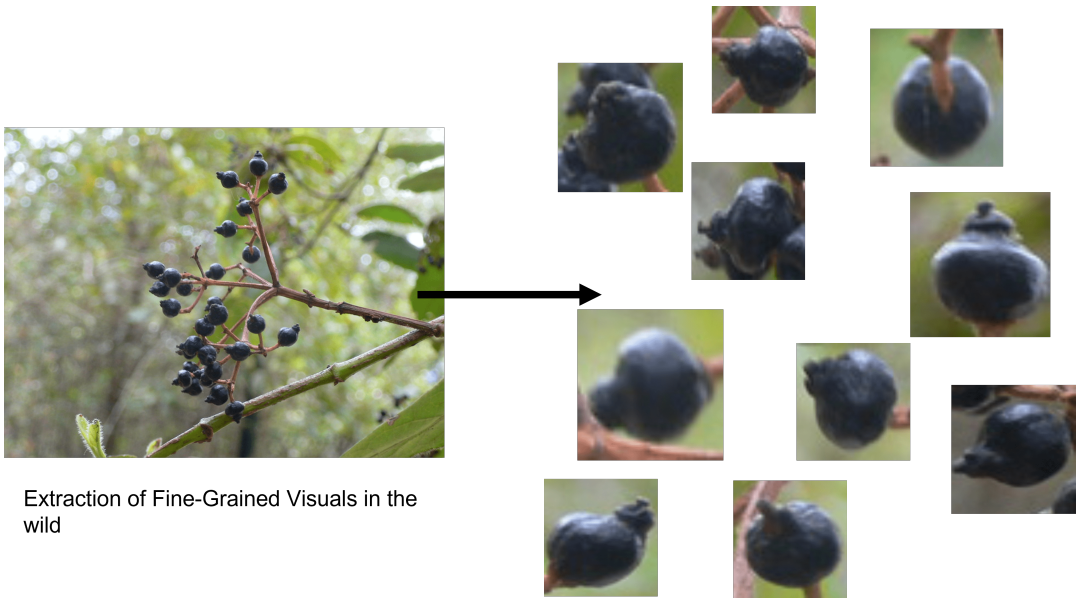


FIGURE 1.1: Fine-Grained Visual.

causing variation. Some successful approach do exist, such as using specialized state-of-art architecture [2] and part based approach such as Cross-X learning [3].

Dealing with data from the wild often has long-tailed distribution. That is number of species considered is very large and the number of examples in the respective species class is very small [4].

Due to the kind of distribution, many suggested approaches tends to lose their effectiveness much quickly. To cope with this variability of the images from the wild and to reduce the background noise. We decided to collect Fine-Grained visuals 'in the wild' dataset with high quality and quantity to predict the species based on the plant organs that are visible to the human eye.

1.3 General Problem Definition

Currently, most of the approaches of plant image analysis deal with identifying leaves or flowers as image subjects, which consists of single leaf or cluster of leaves laid on the white background such are herbarium images which are used for phenological research. Most recently, plant species identification has spread to more difficult and sizable datasets, increasing the number of available tasks in the field. Several competitions conducted by GoogleAI, ImageCLEF, TensorFlow Datasets provide valuable datasets for various analysis tasks including species identification, expanding broader open challenges such as Fine-Grained Visual Classification in the wild. On the other hand, data gathered by citizen

scientists such as iNaturalist, LeafSnap provide a scope for doing tasks in Deep Learning. We have obtained our images from Pl@ntView dataset used in the PlantCLEF 2015 Challenge. These images were gathered as part of a citizen scientist initiative.

1.4 Goal

As mentioned, data obtained through competitions or crowd sourcing carry very high variability. To handle this variability due to fine grained images 'in the wild' dataset, we proposed a three-phase approach. Firstly, we introduce bottom-up approach to plant species identification in the wild that uses object detection as a means of attention to localize the plant organs. This allows us to decrease the data variability by identifying the region of interest while ruling out background noise. Secondly, to introduce this long-tailed dataset that allows training the plant organ detector and plant organ-based species classifier and finally fuse the results obtained in through classifiers. To the best of our knowledge, the dataset we created through this process is the best Fine-Grained Visual Dataset "in the wild" as shown in Figure 1.1. With couple of refinements such as treating the variability in the classes, fine tuning and optimizing the Convolutional Neural Network, we can make use of this dataset for wide range of applications "in the wild". As part of plant analysis task, this report focuses on identifying the plant organ species in the wild and techniques for improving, thereby designing the organ based species classifiers.

1.4.1 Report Contribution

Contributions of this problem report is summarised as follows:

- Dataset is created by manual annotations and curated by ObjectDetector's assisted annotations and manual annotations, thus we have generated a dataset worth 262, 527 samples of organ species.
- From the gathered data, designed organ specific classifier with certain refinements and made use of ResNet as the backbone network for Feature selection

1.4.2 Report Outline

The details of these contributions is as follows, Chapter-2 presents the overall approach of the plant analysis, Chapter-3 discusses the Plant Data gathering and Plant Data Curation, Chapter-4 describes the design and evaluation of the neural network and baseline results thus obtained, Chapter-5 discusses different optimization techniques performed, Chapter-6 presents the comparison of the results obtained, Chapter-7 presents Future work and Conclusions.

Chapter 2

Plant Species Identification

The number of plant species in the wild are extremely huge. Currently as per the records, there are 310,000 plant species in the world [5], [6]. Protecting and having knowledge about these plant species and its respective organs is a difficult task. Also, it not practical for a botanist or an expert to be able to identify and classify all the species. In addition, some plant species are identical to each other and it's really difficult to draw distinction between them. Because of that, recognizing fine-grained images "in the wild" gained attention in computer vision.
Identifying plant images in wild is the sub-component of the Plant-Analysis task. Our plant analysis of organ detection and classification is a three stage bottom-up approach as shown in Figure 2.1. The approach is meant to tackle challenges posed by species identification in the wild, with the fusion technique for aggregation of the plant species prediction from the organ patches.

2.1 Overview of components

2.1.1 Organ detection

During the First stage, we use an object detector to identify and localize the content-types. We have termed these content-types as organs of the plant, such as leaf, flower, fruit, stem and HDL (High-Density Leaves). These content-types are identified as Region of Interest and are passed individually into an organ-based species classifier.

Using object detector to localize the plant organs is advantageous for coping with high variability of the data. Because it selects the information rich regions individually while rejecting the back-ground noise which is not necessary for our identification task. Moreover, down-streaming the organ based species classification and fusion steps will be able to handle a variable number of organs.

For a given image I, we can obtain 'n' regions of interest for our downstream classifier, by deploying an object detector, trained to localize and predict the

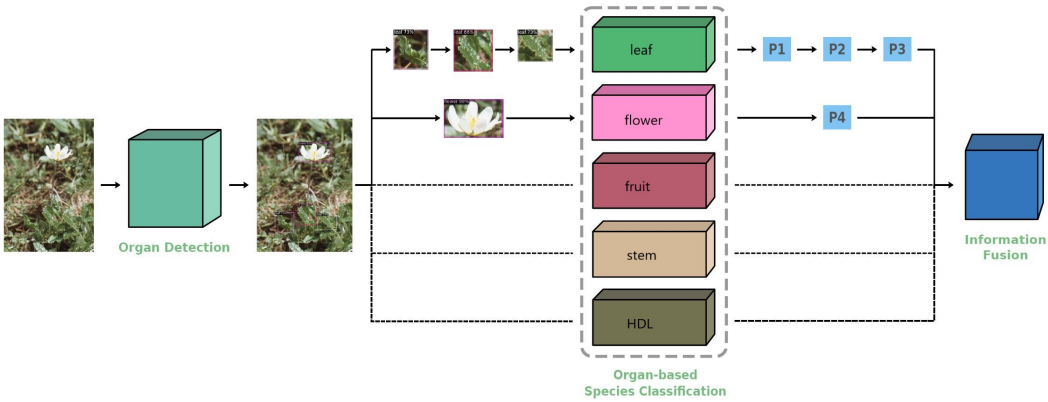


FIGURE 2.1: Approach of plant analysis.

classes of various plant organs. For the i-th ROI, the detector predicts the organ class O_i where o_i belongs to leaf, flower, fruit, stem, HDL. To accomplish this task, we have used Faster-RCNN object detector, built on ResNet-101 backbone.

2.1.2 Object based species classification

From the i-th ROI label by the organ detector o_i, We compute the probabilities of ROI to depict the species s, $p(s \mid o_i)$.We can do this with organ based species classifier. We have used ResidualNetwork (ResNet) as backbone to build the classifier. The design and implementation of classifier and the different optimization techniques that will help to boost the performance of the classifier is going to be discussed in the report.

2.1.3 Information Fusion

The species prediction entails finding the species s that maximizes the probability $p(s)$. We express $p(s)$ as $p(s) = \Sigma_i p(s|o_i)p(o_i)$, where $p(o_i)$ Each of the organ is assumed to be an independent image. A lack of prior knowledge, the natural choice is to assume a uniform prior, which means that $p(s)$, which is equivalent to the sum rule information fusion.

Chapter 3

Plant Dataset Curation

3.1 Data Acquisition

Data is the fuel to do any computer vision task and thus it is more important to have large amount of meaningful data to do any Deep Learning and Computer Vision tasks. Identifying plant based organ species, is a fine grained visual classification task. In order to accomplish the task, it is necessary to have proper sample of fine grained visuals.

As per our knowledge, most of the currently available fine-grained plant datasets in the wild have a few image samples in the respective classes and in order to identify plant organ species we need a dataset of images with high quality and quantity maintaining the variability in the subclasses. For that, we decided to curate a custom dataset.

As part of PlantCLEF 2015 challenge,we have considered Pl@ntNet dataset for our task, it contains 1000 species' images of tree, herbs and ferns based on different types of images belonging to Western European region. Unlike most of the other datasets, which were gathered based on image-centered approach, PlantCLEF dataset is of observation-centered[1] as shown in Figure 3.2, The images we obtained have wide range of objects which are of different scales and are captured from different perspectives. All the images were gathered by 8960 distinct contributors. Each image contributed belongs to seven content-types reported by them as leaf, leaf scan, flower, fruit, stem, branch, entire plant and these images belong to single plant observation. Along with the observation based images, they provided contextual meta-data and social-data(user rating on each image, taxon name etc) provided in a structured XML file associated with each individual image. The meta-data [7], [8]consists of ObservationID: the plant observation ID from which several pictures, associated with the organ, File Name, MediaID, ClassID, Species, Family Name etc. Thus creating a Fine-Grained Plant Organs in the wild dataset with these kind of images will be beneficial and will give a proper meaning to identifying Fine-Grained Plant Organs in the wild. All the images obtained from PlantCLEF are captured under

[1]https://www.imageclef.org/lifeclef/2015/plant

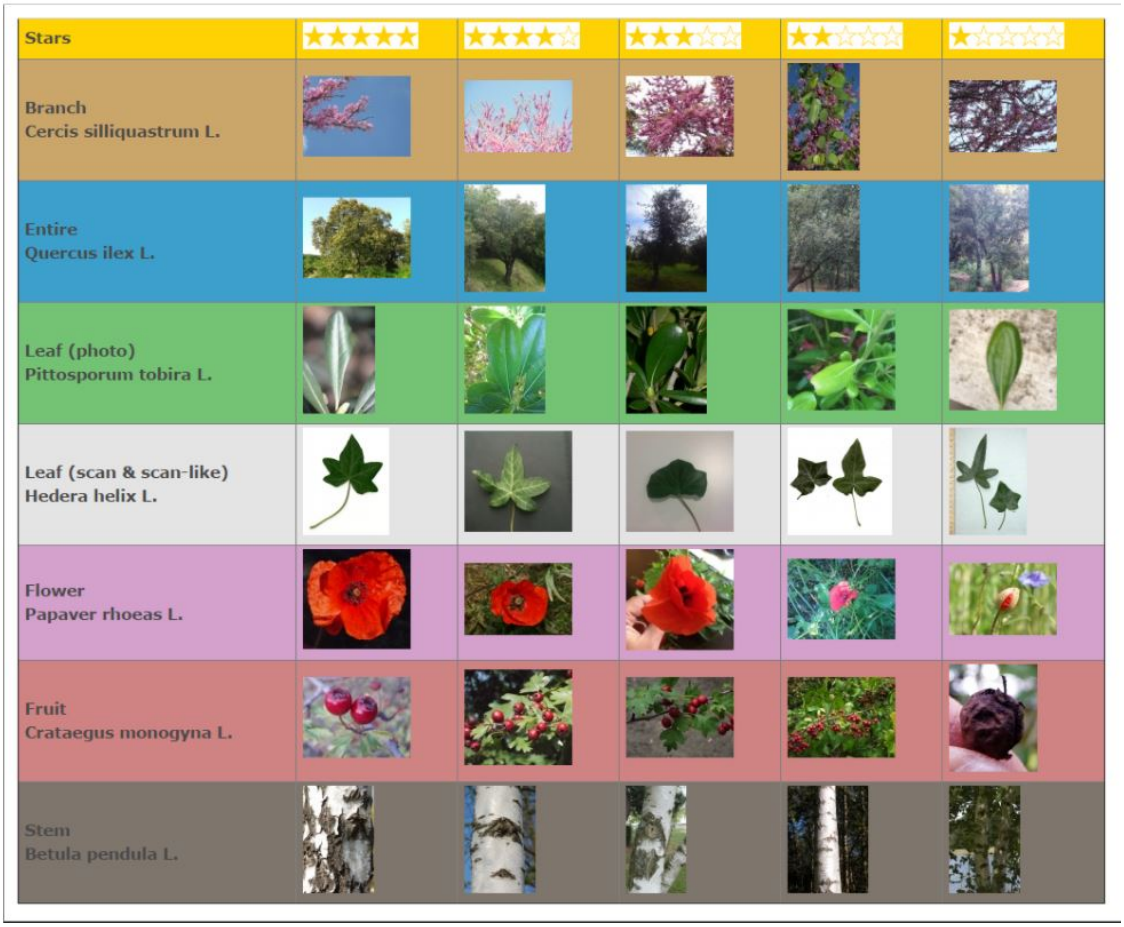


FIGURE 3.1: PlantCLEF collection.

different conditions, and they were ranked as mentioned in Figure 3.1. By considering details of metadata from PlantCLEF2015 dataset, we have narrowed down PlantCLEF content types to the five organs, by combining leafscan and leaf content types to leaf organ, branch and stem to stem organ and have considered Entire as different plant organ and named it as HDL which is High Density Leaves, as it contains high volumes of leaf [7] and considering fruit and flower as is.

We decided to use them as organ labels for our plant species which helps us to identify fine-grained plant organs. We termed our identified organ dataset as DARMA (Detection As Reinforced Means of Attention) dataset [9]. As annotations guide the attention of our network and thus named it as Reinforced. DARMA is comparatively different from the original PlantCLEF dataset as we have selected Regions of Interest using annotations, that is using the labels of the plant-organs, we selected the fine-grained visuals by constructing bounding box on objects of interest ignoring the background information with the help of annotation tool called Snappy-Annotator [10], [11] as shown in Figure 3.4 The dataset thus created by us has high-intra class and low inter-class variability as

FIGURE 3.2: Observation centric.

shown in Figure 3.3.

Initially we started our annotations using traditional annotator called LableImg. Though LabelImg works well for initial annotations, and since we have over million images, we realized LabelImg will not serve our purpose over long run. Thus, to serve the purpose, our plant-analysis team created an annotation tool called snappy-annotator which is created based on Anntool-kit [2].

3.2 Annotations

3.2.1 Snappy-Annotator

Unlike traditional annotators, snappy-annotator makes our life easy. Using this, we can annotate images by hovering our cursor through the region of interest and we need to create a box over the object. The box thus generated is called bounding-box as shown in Figure 3.4. It stores the coordinates of the object called bounding-box coordinates into a structured XML file, which is in the format of Pascal Visual Object Classes. For a given image, it stores the bounding box coordinates (Xmin-top left, Ymin-top left, Xmax-bottom right, Ymax-bottom right) of each object, ImageID, Imagename and size $(H \times W)$.

To be precise during this process, we have collected 113,205 images and annotated 91,761 training images and 21,444 as test images making use of these annotations, the species name and ImageID available in the metadata provided by the PlantCLEF dataset, we extracted the Regions of Interest of these images, which are plant-organs.

3.3 Phases of Data Curation

Our task of gathering data is done in two phases.

[2]Thankyou Stanislav Podgoski and Matthew Keaton

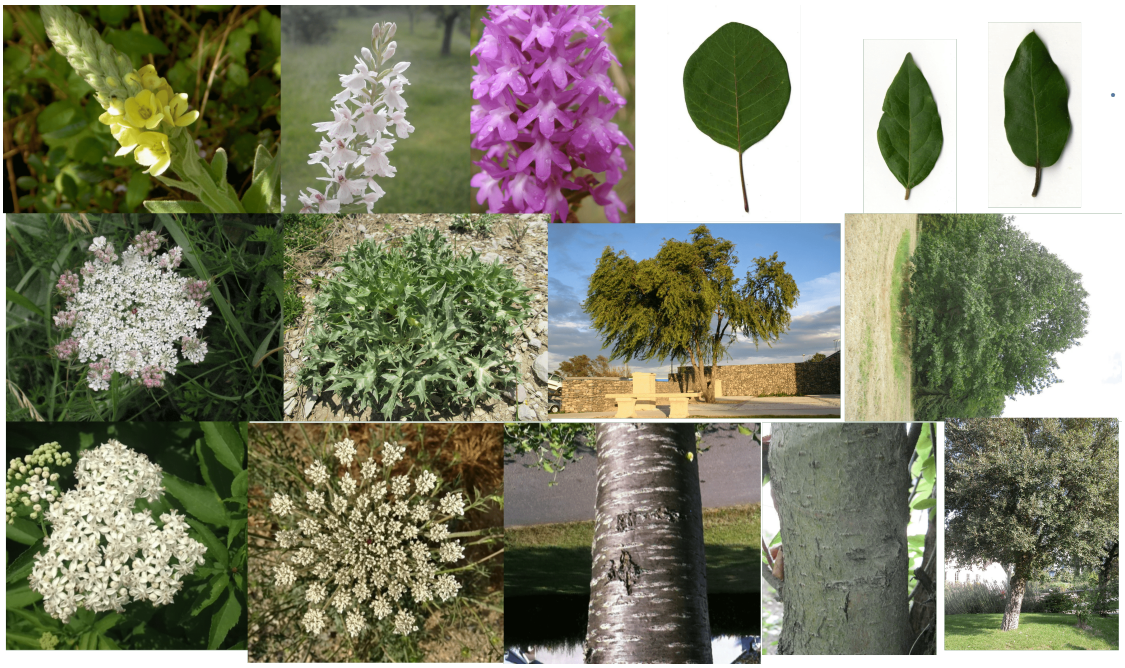


FIGURE 3.3: Intra-Inter class variability.

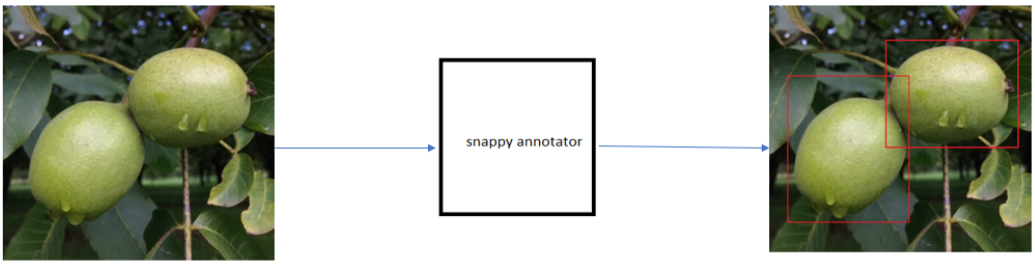


FIGURE 3.4: Annotating the plant organs.

3.3.1 Phase-I

During the Phase-I, we have annotated all the 91, 761 training images from PlantCLEF dataset by beta-version of snappy annotator. We annotated these organs manually by obtaining the species-specific information from the metadata file.

In this phase, annotator provided details of the total annotated images and the current file name, metadata such as species name, selected organ class which is leaf, stem, flower, fruit, or HDL. Annotating all these images is a challenging task as it costs a lot of time and energy. Since the dataset is large we as a group of three, split the dataset equally and completed Phase-I annotations in a span of three months and have gathered plant organs as shown in Table 3.1.

Once the organ information is obtained, we trained a organ detector using Faster-RCNN and ResNet101 as backbone and obtained a set of annotation proposals. We have also trained the classifier with ResNet18 and found that the results obtained as quite confusing. The annotations proposed by object detector are not as we expected. For instance, detector detected fruit instead of flower, flower instead fruit, fruit instead stem, HDL instead stem it also detected certain organs in the HDL region of plant. The Table 3.2 displays the statistics of plant organ species.

We realised that there is a human error while annotating the images. Since a group of three worked on annotations, there was a discrepancy in obtaining the expected results. To avoid human error and also to reduce the labor in accomplishing the task, we decided to go on a Phase-II of annotation check.

TotalCount	262527

TABLE 3.1: Count of annotations in Phase-I.

3.3.2 Phase-II

As mentioned in previous section, To avoid the complications with the dataset created and to enhance the models performance, it is necessary to have a dataset not only with high quantity but also with high quality. Our goal during this

Statistics of number of organs per species			
Organ	Mean	Standard Deviation	Maximum
Leaf	119	219.7	3568
Flower	82.7	67	606
Fruit	30.7	66.1	1122
Stem	4.4	14.6	153
HDL	5.2	7.8	91

TABLE 3.2: Statistics of organ species.

phase is to provide enough qualified samples to identify the organ species. To go through this process, we decided to root around the annotations. In the second phase we processed the annotations in three passes to avoid human errors, we have ensured that only a single person should curate the data in a respective phase as shown in Figure 3.5.

Pass-1

Before narrowing down to the first pass, we have updated the annotation tool. It works in a way that if we pause annotating the image at a particular point and closed the tool, unlike other annotation tools it will display the image from where we stopped. Thus giving us lots of cost savings on time and energy. In the first pass, with the help of the updated tool and considering the annotated data in Phase-I, we cross-checked every image in the dataset and ensured that all the human identifiable organs are annotated. We created the bounding-box on every organ which is identified by the human eye. We have updated the annotations of grouped organ images to individual organs.

Likewise increasing the count of particular organ species in the wild and provided a proper meaning to the fine-grained visualization and also updated the flower organ as flower and fruit organ as fruit respectively. Usually, we find this discrepancy in the family of pine trees. Since the life cycle of the cones is unclear as both flower and fruit of the cones look alike. We have removed the actual stems and annotated only barks or trunks. As most of the dataset contains barks rather than stems and having stems in the dataset would give discriminatory information, we have updated the grouped organs as single organs.

Bridge-Pass

This phase acts like a bridge between Pass-1 and Pass-2. Here, all the manually annotated images are passed to the detector, where it detects the organ specific data along with the imageID. The detector will then generate the OD file, which contains details about the imageID, rank of the object, bounding- box coordinates.

Pass-2

By down streaming the outputs obtained from the object detector proposals, we did a re-check on all the annotations to ensure that detector is performing efficiently on the dataset. If there are no annotations on the organs, the detector generates new ROIs and respective labels on the objects of interest.

In this pass, we ensured that the detector detected the correct organ. In case if it improperly detected an organ with the other organ, we renamed it. If there are any overlapped annotations, then we ensured that the organ is properly annotated with only one single annotation. Making use of the updated annotator,

FIGURE 3.5: Phase-II.

considering the updated criteria to annotate, Unlike in the first phase, we completed the Phase-II of annotations in a month.

Once the plant images are annotated, we pre-processed them by extracting the plant organ species images to obtain fine-grained plant organ specific dataset and hence curated the "DARMA" dataset which is first of its kind and is made publicly available[3]. The Dataset provided by PlantCLEF aims to produce a list of relevant species for each observation, which implies that for a particular plant image they aim to produce one to many pictures related to that particular event. On the contrary, we made use of the PlantCLEF dataset to fine-grain those images to the relevant plant organ species category. As we decided to consider only bark portion of tree, the Table 3.3 shows the increase in count of organ samples on all the organ classes, except stem organ. Going forward we decided to rename stem as bark and make use of the in future analysis work.

Phase	leaf	HDL	bark	fruit	flower	total
Phase-I	107494	5641	5180	20647	72936	211898
Phase-II	130508	5738	4625	32714	88942	262527

TABLE 3.3: Count of annotated images in Phase-I and Phase-II.

[3]https://github.com/wvuvl/DARMA

3.3.3 Uniqueness

Although there are multiple datasets available for plant image analysis in the wild, our approach of collecting the dataset is comparatively different from other classification techniques for a few reasons

1. We annotated the human eye identifiable fine-grained visuals in the wild and thus stored the information of respective organ species data in a structured XML.

2. Our dataset is comparatively larger than many of the publicly available datasets.

Chapter 4

Organ based Species Classification

Once we have gathered the data, labeled the organs of the species, it is necessary to feature engineer and train the model with the dataset. Our aim to identify the plant organs in the wild is a fine-grained visual classification task that is our focus is to differentiate the images, that are hard to distinguish. In order to accomplish the task and to classify the images, it is necessary to design and evaluate classifier.

4.1 Convolutional Neural Network

The agenda of field of computer vision is to enable machines to view the world as humans do and perceive it in a similar manner. The advancements in computer vision and deep learning has constructed an algorithm called Convolutional Neural Network. Image classification aims to connect an image to a set of class labels. It is a supervised learning problem, where a set of pre-labeled training data is fed to a Convolutional Neural Network.

A Convolutional Neural Network is a deep learning algorithm, which takes input as image, assign weights and biases to various objects of the image and be able to differentiate it from one another. The pre-processing required to the

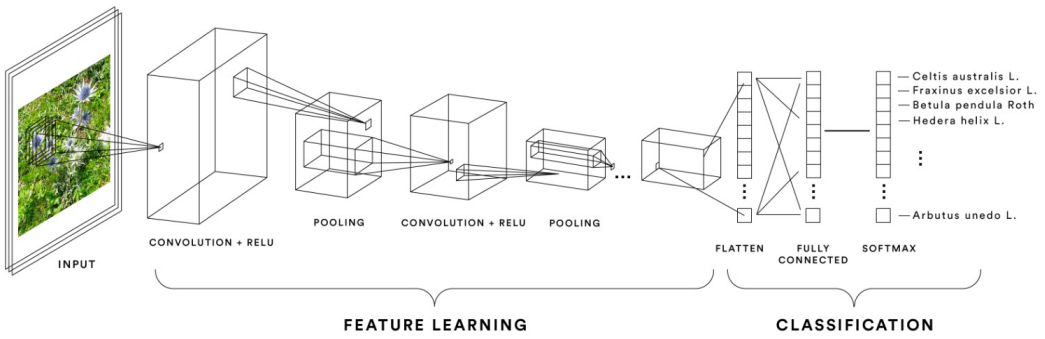


FIGURE 4.1: Basic CNN.

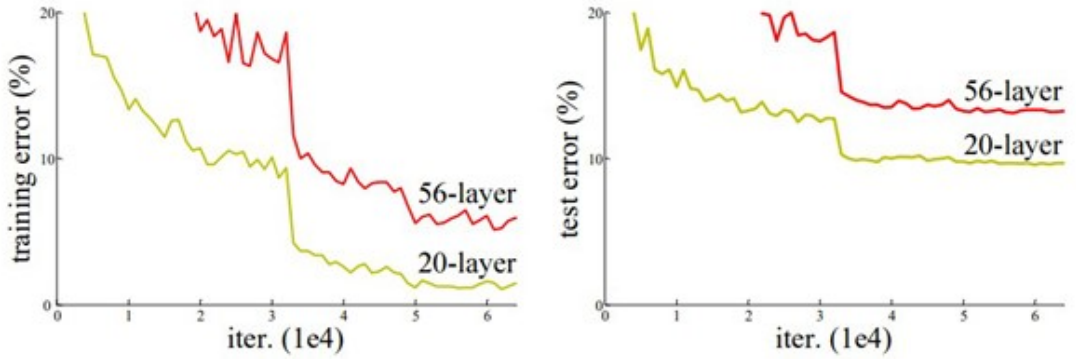


FIGURE 4.2: Training and testing errors for the ResNet Architectures with 20 and 56 layers.

Convolutional Networks (ConvNets) is much less compared to the other classification algorithms, as ConvNets has the ability to learn the model easily. The role of this network is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. It is important to design an architecture which is not only good at learning features but also scalable to massive datasets.

Over the years researchers tend to make deeper neural networks to solve complex tasks to improve the classification accuracy. It is observed that, as we add more layers to the neural network, it becomes difficult to train them and the accuracy starts saturating and at some point degrades too. To solve this kind of problem, ResNet comes to rescue which is the reason why we wanted to build our classifiers using ResNet as the backbone.

Mostly, in order to solve a complex problem, we stack multiple layers in the deep neural networks which results in improved accuracy and performance. The intuition behind adding more layers is that these layers progressively learn more complex features. For instance, for a given input images if the first layer detects the edges, second layer identify the textures, third layer detects the important features, fourth layers detects the objects and so on. It was found that there is a maximum threshold for the depth with the traditional Convolutional Neural Network [12].

We can see from the Figure 4.2 that error percent is more for 56-layers when compared to the 20-layer network in both training and testing cases. This clearly states that adding more layers on top of the network degrades the performance. This could be due to many factors such as initialization of network, optimization function and most importantly vanishing gradient problem.This problem of training the deep networks has been alleviated with the introduction of Residual Networks.

Before deep diving into the Residual Networks, it is necessary to understand certain definitions of CNN and the architecture of a traditional classifier Figure 4.1 and purpose of certain functions associated with it.

4.2 Definitions

4.2.1 Convolutional Neural Networks

For a given input image, the CNN defines a weight matrix and performing convolution operation to extract specific features from the image without losing the information about its spatial arrangements.CNN uses kernel/filter to extract the features.

Advantages of CNN

While solving an image classification problem using traditional Neural Networks has its own problem. Prior to training, it is necessary to convert the 2-Dimensional image to 1-Dimensional. This leads to increase in number of trainable parameters with an increase in size of the image.

One important problem is vanishing gradient problem. This problem is associated with backpropagation algorithm, where weights of neural network are updated through backpropagation. So, in case of deep neural networks, the gradient explodes as it propagates backwards. The reason is traditional Neural Networks, cannot capture sequential information of input image, where are CNN learns the filters automatically without mentioning explicitly. Filters help in extracting the right and relevant features from the input data.

CNN captures the spatial features of the image, follows the concept of the parameter sharing, as a single filter is applied across different parts of an input to produce a feature map.

4.2.2 Working nature of a neural network

A Convolutional Neural Network is composed of various convolutional and pooling layers.

1. The input image is passed to the first convolutional layer and the output thus obtained is an activation map or a feature map. The filters applied in the convolution extract relavant features and pass further.

2. Each feature is thus aided to the class prediction.

3. Pooling layers are added to reduce the number of features made and thus increases the frame count.

4. Fully connected layers form the last few layers in the ConvNets. The input to the fully connected layer is the output from the final pooling or convolutional layer, which is flattened.

5. Activation function such as ReLU is applied for all hidden layers, thus followed by a softmax for the final layer.

4.2.3 Softmax function

It is an activation function that returns a vector of K real values into a vector of K real values that sums to 1. The input values can be a postive, negative, zero or values greater than one. But the output value will be between 0 and 1, so that they can be interpreted as probabilities.

Softmax function can also be termed as softargmax function or a multi-class logistic regression as it is a generalization of logistic regression that can be used for multi-class classification. Its formula is similar to sigmoid function. It is used in classifiers only when the classes are mutually exclusive.

$$\sigma(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

4.3 Residual Networks

ResNet is a specific type of neural network introduced in 2015 [12]. With the help of ResNets, highly complex networks with around 1000 layers can be trained. ResNets are made up of Residual Blocks. It introduces the concept of skip connections and heavy use of batch normalization.

From the Figure 4.3 we can notice that there is a connection which skips some layers, which is a skip connection and is the core of the residual block. Due to the skip connection, the output of the layer is not same as the expected output. Without the skip connection, the input x gets multiplied by the weights of the layer followed by adding a bias term. And then goes through the activation function $f(x)$, where we obtain a new output called $f(x)+x$ When we are following this approach, we may face a problem. Where the dimensions of the input vary from that of the output which can happen with convolutional layer and pooling layer. When dimensions of $f(x)$ is different from the identity mapping x. We can consider two approaches; the skip connection is padded with extra zeros to increase the dimensions or projection method is used to match the dimensions which is done by match and then adding 1x1 conventional layers to input, can be understood from Figure 4.4

$$Y = f(x)+Wx$$

The skip connection in ResNet solves the problem of vanishing gradient in deep neural networks by allowing the gradient to flow through the alternate path. The other way that these connections help is by allowing the model to learn the identity functions which ensures that the higher layer will perform at least as good as the lower layer as shown in Figure 4.3.

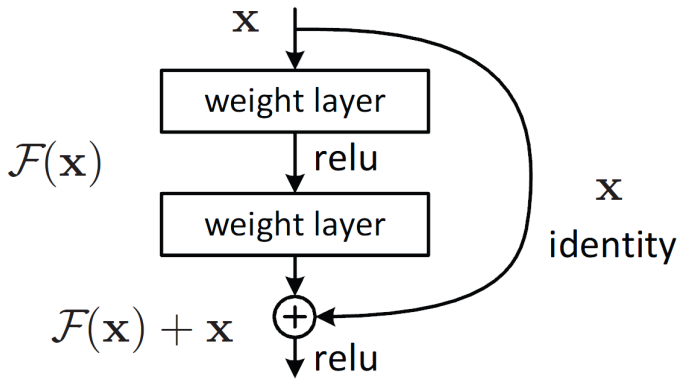


FIGURE 4.3: Skip connection diagram.

4.3.1 Architecture

ResNet-18, a convolutional neural network that is trained on more than a million images from the ImageNet dataset. There are 18 layers present in ResNet-18 architecture, is very useful and one of the efficient neural networks to do image classification and can classify images into 1000 object categories. The network has an input size of 224 × 224.

The architectural Figure 4.5 displays the configuration of layers in ResNet-18. First, there is a convolution layer with 7 × 7 kernel size and stride 2. After this there is a beginning of skip connection. The input from here is added to the output that is achieved by 3x3 max pool layer and two convolution layers with the kernel size 3×4, 64 kernels each. Which makes the first Residual Block.

From here, the output of the residual block is added to the output of two convolution layers with kernel size 3 × 3 and 128 filter. Which makes the second block. Then the third residual block involves the output of the second block through skip connection and the output of two convolution layers with filter size 3 × 3 and 256 such filters. The fourth and final residual block involves output of third block through skip connections and output of two convolution layers with same filter size of 3 × 3 and 512 such filters.

Finally, average pooling layer is applied on the output of the final residual block and received feature map is given to the fully connected layers followed by the softmax function to receive the final output. The output of each layer is shown in the diagram.

The Figure 4.6 shows a pictorial representation of the Resnet-18, excluding the 1X1 Convolutional layer, there are 4 convolutional layers in each Residual module. Together, with the first 7 × 7 convolutional layer and final fully connected layer, there are 18-layers in total. Thus making a 18-layer ResNet.
By adding more layers or by configuring the different number of channels and residual blocks, we can create different residual models, such as deeper ResNet layers. It is easier and simple to modify the ResNet structure. All these factors

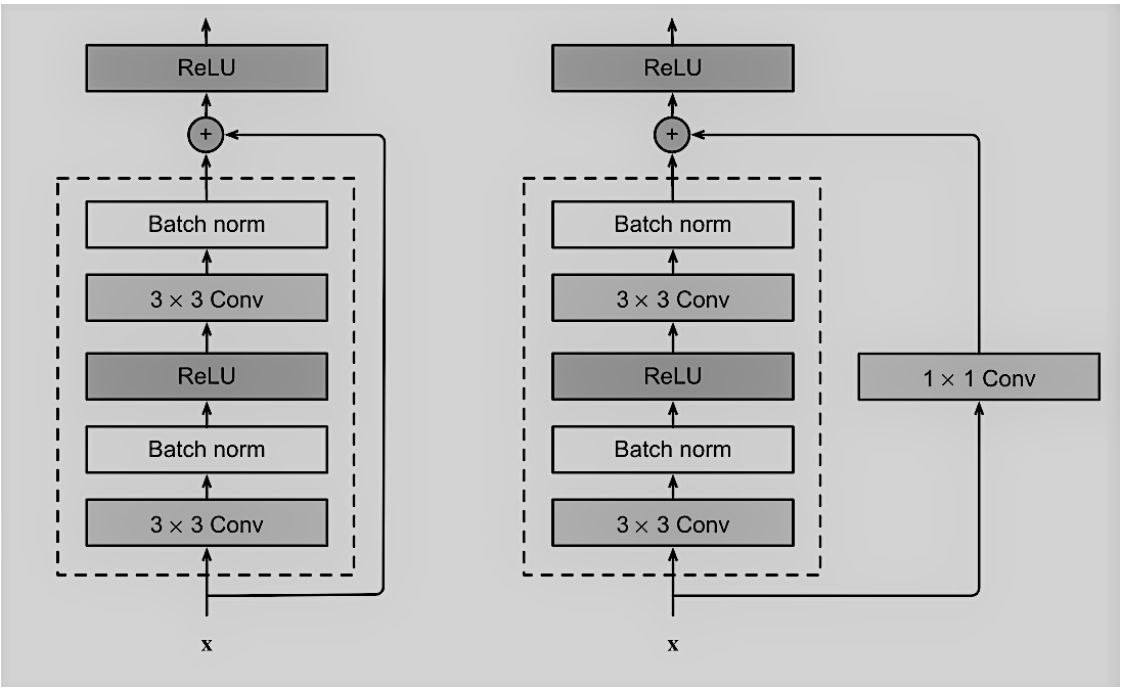


FIGURE 4.4: Skip connection with and without 1×1.

made us to start our plant organ classification using ResNet model.

4.3.2 Implementation

Now that we learnt the working and advantages of ResNet, we have to make the neural network work. Data is the fuel for the working of neural networks. Before feeding the data to the network, it is important to structure the dataset in a way the network accepts it. This process is called pre-processing.
During the detection phase, For a given input image I, we obtain n-region of interests where we pass the ROIs to the organ based classifiers by deploying the object detector, training to localize and predicting the classes of various plant organs. Thus the detected organ species is fed as input to the respective organ classifier as shown in Figure 4.7 and probability of the organ class is identified. We perform organ classification separately for respective organ classes as shown in Figure 2.1 means, we build leaf classifier, fruit classifier, flower classifier, stem classifier and HDL classifier separately and fuse the obtained results to get the overall accuracy by using sum rule.

In order to perform organ based classification, we have used pre-trained ResNet18 which was trained on ImageNet dataset and later for comparative study performed our experiments on Pretrained ResNet34, ResNet-50 respectively. We also performed a baseline comparison of accuracy on ResNet101.

Layer Name	**Output Size**	**ResNet-18**
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

FIGURE 4.5: ResNet-18 architecture.

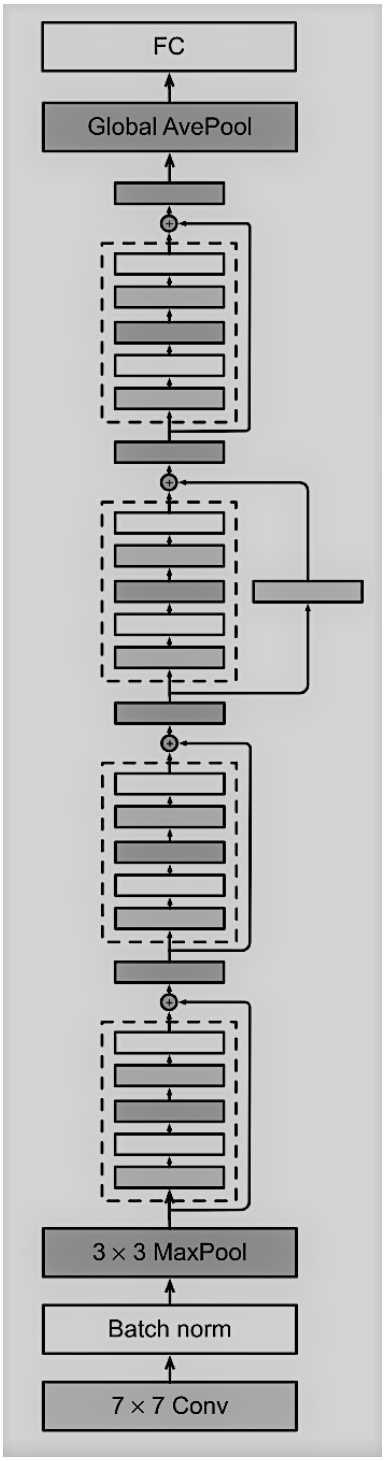


FIGURE 4.6: Block diagram of ResNet-18 with skip connection.

For each organ species, the images were split into 70% for training, 10% for validation and 20% for testing, since we have a long-tailed distribution of dataset. When fewer than 10 images were included for species, we ensured that atleast one sample was placed into the validation and test set. Although there are 1000 classes we have reduced our focus of study to 161 species of plants ensuring that each and every organ class (species) has atleast one sample of image as we took reference of leaf organ which has atleast 130 samples in all of its classes. Table 4.1 represents the Count of 161 Organ Species that we used for our Phase-II evaluation, and the number of samples in each train, test and validation split.

	161 Species Statistics				
	leaf	flower	fruit	stem	HDL
train	45955	9853	11430	2289	1220
test	12992	2714	3073	613	380
valid	6248	1446	1590	343	188
Total	65195	14013	16093	3245	1788

TABLE 4.1: Split count of 161 organ species.

Our Plant organ species classifier is built on PyTorch 1.7.1 framework, CUDA Version: 11.2, GPU: TITAN X Pascal. The annotated images that are used as input are of different scales varying from 69×800 to 800×800. We have resized the input image dimensions to 224×224,which is the suggested input dimension of Residual Network. If the size of the image is large, it gets shrinked to 224×224 in the later layers of network. If the input dimensions are less, PyTorch implements biliniear interpolation on input image. So to avoid such discrepancies, we have resized the input images before training.

It is best practice to normalize the input images in the range of values prior to feeding them to the model. Normalizing the input images is important because one image might have very high pixel value while others having very lower pixel value. When we re-scale the images, we ensure that each image is equally contributing to the cost function. Since we transfer the learning model, which is trained on ImageNet dataset, we normalize the input images with mean and standard deviation of $(0.485, 0.456, 0.406)$, $(0.229, 0.224, 0.225)$ respectively. We want to train the algorithm to be as invariant as possible and thus improve the ability to generalize the train data. Since we want our algorithm to perform well on recognizing and classifying the images from different perspective and to increase the variance, we performed augmentations such as Random HorizontalFlip, Random VerticalFlip and Random Rotations. We used default Adam optimizer and initial learning rate of 1e-4 is used to train with mini batch size of 32. The maximum number of iterations is for 144000 for leaf dataset and takes approximately 11 hours to train a particular leaf organ classifier, while the least

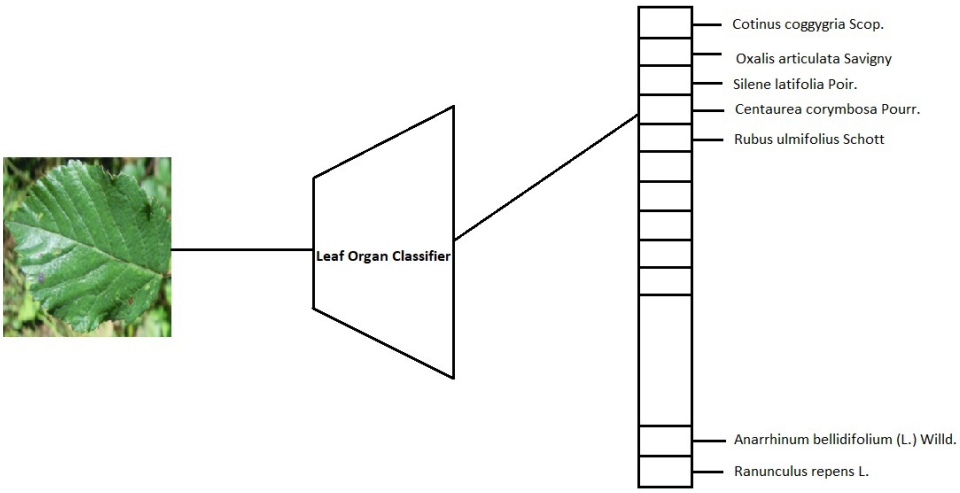


FIGURE 4.7: Species classifier based on Leaf organ.

is for HDL (High Density Leaf).

Baseline	ResNet18	ResNet34	ResNet50	ResNet101
leaf	68.7576	70.2432	72.0751	72.6447
fruit	61.8613	62.0240	65.2782	65.3107
flower	74.392	74.8710	79.2925	75.6816
stem	58.7275	61.3376	61.0114	58.4013
hdl	37.8947	36.0526	39.2105	37.6315

TABLE 4.2: Baseline results of 161 organ species.

The Table 4.2 displays accuracy of the baseline classifier ResNet models. For comparative study, we have used a learning rate of 0.0001 for all the classifiers. From the obtained results, we can understand that we cannot compare the accuracy of organs, as there is class-imbalance in the respective organ species. Comparatively leaf and flower has more samples than Stem or HDL, for that reason Leaf and Flower classifiers achieved better results in deeper Residual models such as ResNet101 and ResNet50, where as Stem, HDL performed better on ResNet-34. To understand more about their performances, we performed multiple experiments with certain training refinements which is discussed in further chapters.

Chapter 5

Performance Optimization

Most often when we train our Neural Network, we may not be happy with the obtained results. We seek to increase the model performance by performing certain effective training tricks.

Much of the recent progress made in image classification research can be credited to training procedure refinements, such as changes in data augmentations and optimization methods.Optimizing the model in most cases work by hyperparameter tuning or by implementing certain tricks and techniques [13]. In the report we cover mostly the different combination of certain tricks and how they refine the models performance and evaluate empirically their impact on the dataset. By applying certain combination of tricks we can certainly say that models performance increase to almost 73% to 65%.

Before discussing about different techniques or tricks that were used to optimize the performance. It is better to know about certain functions and why we are using them for optimization technique.

5.1 Definitions

Optimization

Optimization is the process of adjusting hyper-parameters in order to minimize the cost function by using one of the optimization techniques. It is important to minimize the cost function because it describes the discrepancy between the true value of the estimated parameter and what the model has predicted.

Gradient Descent

Gradient Descent is one of the most popular algorithms to perform optimization and is used as common way to optimize neural networks, that is minimize the objective function $J(\theta)$ where parameterized by models parameters (that is weights). In other words, the procedure of repeatedly evaluating the gradient and then performing a parameter update is called Gradient Descent.

$$\theta^1 = \theta^0 - \alpha \nabla J(\theta) \tag{5.1}$$

where θ shows the position of weights or other parameters, $\nabla J(\theta)$ shows the direction of the fastest increase gradient.

Adam Optimizer

Adaptive Moment Estimation (Adam) is a method that computes adaptive learning rates for each parameter. Adam also keeps an exponential decaying average of past gradients mt, similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a ball with a friction, which thus prefers flat minima in the error surface.

Regularization

Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values. It is used to treat the overfitting problem.

Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also required for some algorithms to model the data correctly.

Loss Function

Loss function/cost function: is a measure of quality of a particular set of parameters based on how well the induced scores agree with the ground truth labels. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

5.2 LR Scheduling

When building deep learning model the most common problem we face is choosing the correct hyper-parameters. In Machine Learning, a hyper-parameter is a configuration variable that's external to the model and whose value is not estimated from the data given.

5.2.1 Learning Rate (λ)

Learning Rate is one such hyper-parameter that defines the adjustment in the weights of our network with respect to the loss gradient descent. It determines how fast or slow we move towards the optimal weights.

For the Gradient Descent to work, we must set the learning rate to an appropriate value. This parameter determines how fast or slow we will move towards the optimal weights. If the learning rate is very large, we will skip the optimal solution. If it is too small, we will need too many iterations to converge to the best values. Tuning the learning rate, is most important hyper-parameter for tuning neural networks. A good learning rate could be the difference between a model that does not learn anything and a model that presents state-of- the-art results. Learning rate schedules seek to adjust the learning rate during training by reducing the learning rate according to a pre-defined schedule.
Before implementing the techniques,we performed certain experiments on Flower organ to decide the optimal Learning Rate and Batch-size to build and train the classifier. Figure 5.3 represents the comparative results of different Learning Rates. All the Learning Rates except 1e-4 gave underfitting or overfitting results.

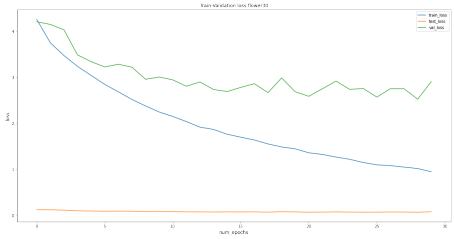
5.2.2 Constant Learning Rate

Constant Learning rate is the default learning rate schedule, where we have used Adam as gradient descent optimizer. Moment and decay rate both are set to zero by default. It is tricky to choose the correct learning rate the performs better. By experimenting with different range of learning rates from $1e-1$ to $1e-5$ as shown in Figures 5.1 and 5.2, we obtained better loss curve while experimenting with 0.0001 as shown in Figure 5.3 and thus we considered Learning Rate value 0.0001 an ideal learning rate as it shows a relative good performance to start with and thus served as a baseline for us to experiment with techniques. All the experiments to choose learning rate are conducted on Flower organ dataset,. This method of improving the convergence rate of hyper-parameters reduces the need for the manual tuning.

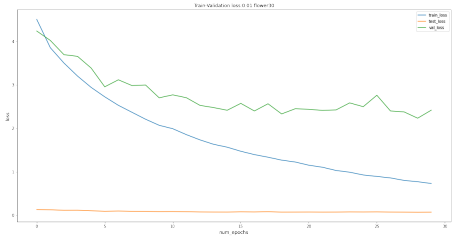
Baseline results: Since all the organ species are trained on a single Learning Rate, with no refinements. We consider Table 4.2 results as baseline accuracy results.

5.2.3 Multi-Step Learning Rate

Multi Step decay drops the learning rate by a factor of gamma (generally we use 0.1) once the number of epochs reaches one of the milestones where gamma is the hyper-parameter for tuning the Learning Rate. Table 5.1 displays the results of Multi-Step Scheduling, with initial Learning Rate as 0.0001 The numbers say

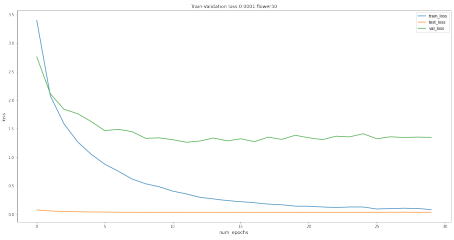


(A) LR=0.1

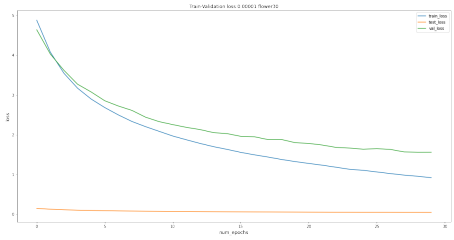


(B) LR=0.01

FIGURE 5.1: LR=0.1 and LR=0.01 with respect to loss curve.



(A) LR=0.001



(B) LR=0.00001

FIGURE 5.2: LR=0.001 and LR=0.00001 with respect to loss curve.

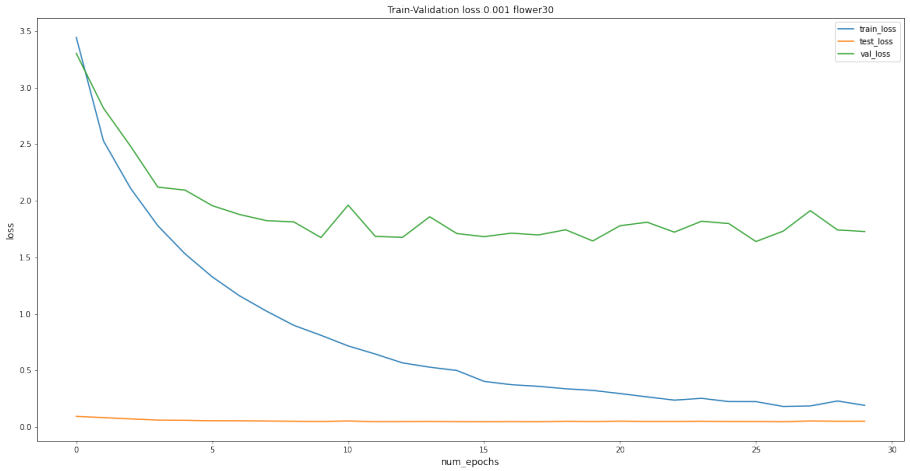


FIGURE 5.3: Ideal LearningRate: 1e-4.

that accuracy will definitely improve by adoptung multistep scheduling of the Learning Rate. We can see improvement on all Classifiers .

MultiStep	**ResNet50**	**ResNet34**	**ResNet18**
HDL	40.789	39.474	39.211
BL	39.211	36.053	37.895
Change	**1.579**	**3.421**	**1.316**
Stem	63.785	61.011	62.969
BL	61.011	61.338	58.728
Change	**2.773**	-0.326	**4.242**
Fruit	68.630	67.654	65.571
BL	65.278	62.024	61.861
Change	**3.352**	**5.630**	**3.710**
Flower		77.634	75.240
BL	79.293	74.871	74.392
Change		**2.763**	**0.848**
Leaf	81.614	80.324	79.256
BL	72.075	70.243	68.758
Change	**9.539**	**10.081**	**10.498**

TABLE 5.1: Accuracy comparison after adopting MultiStep Learning Rate.

Cosine Scheduling	**ResNet50**	**ResNet34**	**ResNet18**
HDL	39.474	38.947	37.632
BL	39.211	36.053	37.895
Change	**0.263**	**2.895**	-0.263
Stem	64.111	61.501	61.501
BL	61.011	61.338	58.728
Change	**3.100**	**0.163**	**2.773**
Fruit	67.816	67.686	65.636
BL	65.278	62.024	61.861
Change	**2.538**	**5.662**	**3.775**
Flower	81.614	80.324	79.256
BL	79.293	74.871	74.392
Change	**2.321**	**5.453**	**4.864**
Leaf	72.529		72.037
BL	72.075	70.243	68.758
Change	**0.454**		**3.279**

TABLE 5.2: Accuracy comparsion after adapting Cosine Scheduling with initial LR: 0.0001, T=total batchsize.

5.2.4 Cosine Annealing

Cosine scheduling [13] relies on the observation that we might not want to decrease the learning rate too drastically in the beginning, moreover, we might want to refine the solution in the end using very small learning rate. This results in a cosine like schedule with the function form for learning rate in the range

$$\eta_t = \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)\eta \tag{5.2}$$

where η_0 is the initial learning rate, η_T is the target rate at time, Furthermore, for T, we simply pin the value to η_T without increasing it again. In most of our experiments, we set the max update step to 80 for ηT Decreasing the learning rate during training can lead to improved accuracy and reduce the over fitting of the model. Optimization serves multiple purposes in deep learning. Besides minimizing the training objective, different choices of optimization algorithms and learning rate scheduling can lead to different amounts of generalization and overfitting on the test set(for the same amount of training error) Accuracy achieved through Cosine Scheduling is shown in Table 5.2 and we can clearly say that compared to BaseLine accuracy, there is a gradual improvement in all the organs performance. We can say from the results, it performed well on ResNet34 and ResNet50.

FIGURE 5.4: MixUp combination.

5.3 MixUp

Though we discussed about data augmentation in the feature engineering stage and performed certain augmentation on input data before passing them to the model, now we are going to consider another data augmentation technique called MixUp. MixUp is a simple learning principle used to alleviate the undesirable behaviors such as memorization and sensitivity in deep neural networks. MixUp trains on a convex combination of pairs and labels.

With MixUp [14], we can create virtual training examples to enlarge the support of the training distribution - especially when you lack a large dataset - without incurring high computational costs.

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda)x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda)y_j \end{aligned} \tag{5.3}$$

where (x_i , y_i) and (x_j , y_j) are two feature-target vectors drawn at random from the training data, and $\lambda \in [0,1]$. The MixUp hyper-parameter α controls the strength of interpolation between feature-target pairs. Figure 5.4 shows a sample MixUp combination of two images

Benefits of MixUp include (as described in the paper) robustness to adversarial examples and stabilized GAN (Generative Adversarial Networks) training. This technique is really good at regularizing the ML models. There are several data augmentation techniques that extend MixUp such as CutMix and AugMix. From the Table 5.3, we can clearly say that there is no much improvement in accuracy as MixUp creates the noisy labels. It can be used as the effective training technique, rather than for improvising the results.

5.4 Label Smoothing

Several researchers noted that better classification performance and faster convergence could be attained by performing gradient descent to minimize cross entropy even in early days of neural network research, there were indications that other, more exotic objectives could outperform the standard cross entropy

MixUp	**ResNet50**	**ResNet34**	**ResNet18**
HDL	39.474	37.368	35.789
BL	39.211	36.053	37.895
Change	**0.263**	**1.316**	-2.105
Stem	61.175	-	56.770
BL	61.011	61.338	58.728
Change	**0.163**	**-**	-1.958
Fruit	63.814	63.846	61.048
BL	65.278	62.024	61.861
Change	-1.464	**1.822**	-0.813
Flower	77.598	75.350	76.013
BL	79.293	74.871	74.392
Change	-1.695	**0.479**	**1.621**
Leaf	73.376	71.005	70.651
BL	72.075	70.243	68.758
Change	**1.301**	**0.762**	**1.894**

TABLE 5.3: Accuracy comparision after implementing MixUp.

loss. More recently, Szegedy et al [15] introduced label smoothing, which improves accuracy by computing cross entropy not with the “hard" targets from the dataset, but with a weighted mixture of these targets with the uniform distribution.
Generalization and learning rate of a multi-class neural network can be improved by weighted average of hard-targets thus obtaining soft-targets.
The role of regularization is to modify a deep learning model to perform well with inputs outside the training dataset. Specifically, regularizing the focus on reducing the test or generalization error without affecting the initial training error.By artificially softening the targets, label smoothing prevents the network from becoming over-confident.

We predict the distribution of neural network as a function of the activations in the penultimate layer as p_k

$$p_k = \frac{e^{\mathbf{x}^T \mathbf{w}k}}{\sum_{l=1}^{L} e^{\mathbf{x}^T \mathbf{w}_l}} \tag{5.4}$$

where p_k is the likelihood of the model assigns to k^th class,w_k represents the weights and biases of the last layer, x is the vector of activations of penultimate layer. For a network trained with hard targets, we minimize the expected value of the cross-entropy between the true targets y_k and the networks output p_k as

LabelSmoothing	**ResNet50**	**ResNet34**	**ResNet18**
HDL	39.211	36.053	39.474
BL	39.21053	36.05263	37.8947
Change	0.000	0.000	**1.579**
Stem	59.706	58.401	59.054
BL	61.01142	61.33768	58.7275
Change	-1.305	-2.936	**0.326**
Fruit	65.018	65.181	64.660
BL	65.27823	62.02408	61.8613
Change	-0.260	**3.157**	**2.799**
Flower	77.634	77.045	76.382
BL	79.29255	74.87104	74.392
Change	-1.658	**2.174**	**1.990**
leaf	73.160	71.259	69.828
BL	72.07512	70.24323	68.7576
Change	**1.085**	**1.016**	**1.070**

TABLE 5.4: Accuracy comparision after applying Label Smoothing.

in

$$H(\mathbf{y}, \mathbf{p}) = \sum_{k=1}^{K} -y_k \log(p_k) \tag{5.5}$$

where y_k is 1 for the correct class and 0 for the rest. For a network trained with a label smoothing of parameters α, we minimize the modified targets $y_k^L S$ and the networks' output p_k where

$$y_k^{LS} = y_k(1-\alpha) + \alpha/K \tag{5.6}$$

Comparison of Label Smoothing Accuracy with differentt organ species is shown in Table 5.4

Observation from obtained results: All our observations are with the Label Smoothing parameter α=0.1 since there is no much data in HDL, stem there seems to be no much improvement in ResNet18, but not in Deeper Networks such as ResNet34, ResNet50. Results with respect to Fruit dataset were significantly improved in ResNet34 and ResNet18. Though there is data variability, due to more number of samples we can observe improvement in results. Flower classifier performed better on ResNet34 Leaf classifier sees improvement on all the Networks, Since there are more samples of Leaf when compared to other organs, we can see increase in accuracy in all the ResNets. But Significantly boosted the performance in ResNet50.

5.4.1 Knowledge Distillation

Knowledge distillation is a model compression technique whereby a small network is taught by a larger trained neural network. This training setting is sometimes referred to as "teacher-student" where the large model is the teacher and small model is student. Distilling knowledge from a large model into a small one, we can train the small model to generalize in the same way as the large model. If a small model is trained to generalize in the same way will typically do much better on the test data than a small model that is trained in the normal way on the same training set. Transferring the generalization ability of the cumbersome model to a small model is to use the class probabilities produced by the teacher model as "soft targets" for training the small model.
One of the solutions called distillation is proposed to raise the temperature of the final softmax until the teacher model produce a suitably soft set of targets [16]. We then use the same high temperature when training the small model to match these soft targets.
We know that neural network produce class probabilities using a "softmax" layer as output that converts the the logit z_i computer for each class into a probability q_i by comparing z_i with the other logits,

$$q_i = \frac{\exp\left(z_i/T\right)}{\sum_j \exp\left(z_j/T\right)} \tag{5.7}$$

Where T is the Temperature that is normally set to 1. If it is 1 the softmax function acts like regular cross entropy loss. Using higher value for T produces a softer probability distribution over classes.
In the simplest form of distillation, knowledge is transferred to the distillation model by training it on the transfer set and using the soft target distribution for each case in the transfer set that is produced by using the cumbersome model with high temperature in its SoftMax.

Benefits from label smoothing

As mentioned in the previous section, Label Smoothing(LS) [15], [17] is a technique to soften one-hot label y be a factor of ϵ, such that the modified label becomes

$$\tilde{y}_i^{\mathrm{LS}} = (1-\epsilon)y_i + \epsilon/K \tag{5.8}$$

Label smoothing mitigates the over-confidence issue of the neural networks, and improves model calibration. Knowledge Distillation on the other hand, uses an additional teacher model's predictions. Logits gradient for KD is given by:

$$\theta^*_{\mathrm{KD}} = \arg\min_{\theta\in\Theta}\left\{\mathcal{L}^{\mathrm{KD}}(\mathbf{y},\mathbf{p},f(\mathbf{x};\theta),\lambda,T) = (1-\lambda)\mathcal{H}(\mathbf{y},\mathbf{q}) + \lambda\mathcal{H}(\tilde{\mathbf{p}},\tilde{\mathbf{q}})\right\}, \tag{5.9}$$

Where $\lambda \in [0,1]$ is a hyper-parameter and q and p are temperature softened student and teachers predictions. Logits gradient for KD is given by:

$$\partial \mathcal{L}^{\mathrm{KD}} / \partial z_i = (1-\lambda)\left(q_i - y_i\right) + (\lambda / T)\left(\tilde{q}_i - \tilde{p}_i\right). \text{ Lets denote } \partial_i^{\mathrm{KD}} = \partial \mathcal{L}^{\mathrm{KD}} / \partial z_i \tag{5.10}$$

Though results are not produced for this technique, it is necessary to know that implementing Knowledge Distillation will boost the performance accuracy.

Chapter 6

Results

Along with annotating and curating the dataset, One of the most challenging task in this problem report is to fetch results by optimizing the model. After doing hundreds of experiments by fine tuning the model parameters, we can say that by certain combination of tricks and techniques we can effectively achieve good accuracy and boost in the model performance.

Below are the respective results of plant organ species classification. All the input images that are fed to the model are of same size 224×224. We performed training with $1e-4$ as initial Learning Rate.

Schedulers and Label Smoothing: From Accuracy Tables 6.1 and 6.2 we can say that Label Smoothing with Cosine and MultiStep Scheduling boosts the accuracy upto +9% on leaf organ dataset, while reasonable improvement on Flower, Fruit organ. There seems to be pretty good improvement in Multi-Step Scheduling, which shows that step-wise improvement in Learning Rate works better than Relative decrease in the Learning Rate to minimum value.

Schedulers and MixUp Augmentation: We can conclude from the accuracy Tables 6.4 and 6.5 that performing Cosine Learning Rate scheduler on the model will significantly improve the model performance. Performing multiple experiments on MixUp with MultiStep Scheduling will help us know the ideal scheduler to perform MixUp with. Compared to the results obtained through Label Smoothing, we can see there is no significant increase in the performance with MixUp.

MixUp Augmentation and Label Smoothing: As the paper says [18] and also from the results Table 6.3 we can clearly say that there is no significant improvement in the accuracy. Though we can see improvements on certain organs,it is due to over-fitting of the train model. From the table we can see that there is an increase in accuracy in HDL and Fruit organ species as most of the samples in these organ species have similar images. Performing MixUp and expecting increase in accuracy will definitely deviate our performance. The reason is the

labels obtained through MixUp have a noisy values, as most probably the predictions on resultant output will be on the noisy image and its noisy label. So it is not suggested to perform this combination for performance optimization.

MixUp-Label Smoothing with Schedulers: From the Tables 6.7 and 6.6 there is a significant increase in the accuracy in almost all the organ species. We can observe maximum increase in Leaf, Fruit and Flower organ classifier. As there are more samples in these organ species. Also we can see there is no much improvement in HDL as most of the species look similar.

Confusion Matrix Results: As we saw significant improvement in accuracy on all the organ classifiers by using ResNet34 , we have constructed confusion matrices on certsin combination of organ species using ResNet34. We can observe from the Stem confusion matrix Figure 6.5, 6.9 and HDL confusion matrix Figure 6.4, 6.8, 6.12 that accuracy can still be improved by obtaining more samples for the respective species of the organ class. That is most of the classes that are predicted are False positives and False Negatives, we can conclude that though there is an increase in accuracy from baseline to adopting optimization techniques, we cannot rely on these results. From the Leaf confusion matrix Figures 6.3, 6.10 and Flower confusion matrix Figures 6.2, 6.6 and 6.14 we can say that although there are many true positive results adding more samples to the respective organ dataset will improve the accuracy. From Fruit organ classifier's, confusion matrix Figures 6.1, 6.7 and 6.13 though there is improvement in accuracy from baseline results to combination of tricks, we can say that the model performs much better if there are more data samples in respective classes, thus treating the underfitted model, since most of the fruit samples are alike.

Overview of Results: From the obtained results, we can say that model performs well only if there is proper balanced data, that adds meaning to FGVC. Also by using deeper networks we may not get accuracy that we need, there is a high probability that the model Overfits if we increase the layers. A model performs better only if there is data with both quality and quantity and thus choosing the proper model with respect to the number of samples in dataset is important.

Cosine_LS	**ResNet50**	**ResNet34**	**ResNet18**
HDL	39.737	39.474	38.684
BL	39.211	36.053	37.895
Change	**0.526**	**3.421**	**0.790**
stem	60.359	63.132	59.706
BL	61.011	61.338	58.728
Change	-0.653	**1.794**	**0.979**
Fruit	64.009	66.092	63.977
BL	72.075	70.243	68.758
Change	-8.066	-4.151	-4.781
flower	77.487	80.472	76.603
BL	79.293	74.871	74.392
Change	-1.805	**5.601**	**2.211**
Leaf	73.953		72.391
BL	65.278	62.024	61.861
Change	**8.675**		**10.529**

TABLE 6.1: Accuracy comparison after adopting Cosine Scheduling and Label Smoothing.

MultiStep_Label	**ResNet50**	**ResNet34**	**ResNet18**
HDL	38.158	38.421	39.737
BL	39.211	36.053	37.895
Change	-1.053	**2.368**	**1.842**
Stem	61.664		63.295
BL	61.011	61.338	58.728
Change	**0.653**	**0.000**	**4.568**
Fruit	68.858	77.634	
BL	65.278	62.024	61.861
Change	**3.580**	**15.610**	0.000
Flower	81.651		
BL	79.293	74.871	74.392
Change	**2.358**		
Leaf	74.877		
BL	65.278	62.024	61.861
Change	**9.599**		

TABLE 6.2: Accuracy comparison after adopting MultiStep Scheduling and Label Smoothing.

Mixup_label	**ResNet50**	**ResNet34**	**ResNet18**
HDL	37.105	37.105	37.632
BL	39.211	36.053	37.895
Change	-2.105	**1.053**	-0.263
Stem	57.749		
BL	61.011	61.338	58.728
Change	-3.263		
Fruit	64.888	63.033	
BL	65.278	62.024	61.861
Change	-0.390	**1.009**	
Flower	77.155		
BL	79.293	74.871	74.392
Change	-2.137		
Leaf	74.877		
BL	65.278	62.024	61.861
Change	**9.599**		

TABLE 6.3: Accuracy comparison after adopting MixUp and Label Smoothing.

Mixup_Cosine	**Resnet50**	**ResNet34**	**ResNet18**
HDL	38.421	37.632	37.368
Baseline	39.211	37.105	37.895
Change	-0.79	**0.526**	-0.526
Stem	63.458	62.806	60.033
BL	61.011	59.706	58.728
Change	**2.447**	**3.1**	**1.305**
Fruit	67.296	65.05	63.586
BL	65.278	62.024	61.861
Change	**2.018**	**3.026**	**1.725**
Flower	80.361	78.592	72.108
BL	79.293	74.871	74.392
Change	**1.069**	**3.721**	-2.285
Leaf	75.847	73.73	71.436
BL	72.075	70.243	68.758
Change	**3.771**	**3.487**	**2.679**

TABLE 6.4: Accuracy comparison after adopting MixUp and Cosine Scheduling.

Mixup_multistep	**ResNet34**
HDL	31.053
BL	37.632
Change	-6.579
Stem	57.912
BL	62.806
Change	-4.894
Fruit	65.408
BL	59.706
Change	-5.702
Flower	79.108
BL	74.871
Change	**4.237**
leaf	73.276
BL	73.73
Change	-0.454

TABLE 6.5: Accuracy comparison after adopting MixUp and MultiStep Scheduling.

Mixup_Label_Cosine	**ResNet50**	**ResNet34**	**ResNet18**
HDL	39.21	37.63	35.79
BL	38.42	37.63	37.37
Change	**0.79**	0	-1.58
Stem	62.64	60.69	59.38
BL	61.01	61.34	58.73
Change	**1.63**	-0.65	**0.65**
Fruit	66.48	65.67	63.68
BL	65.28	62.02	61.86
Change	**1.2**	**3.64**	**1.82**
flower	79.92	79.26	77.78
BL	79.29	74.87	74.39
Change	**0.63**	**4.38**	**3.39**
Leaf	76.31		72.18
BL	72.08	70.24	68.76
Change	**4.23**		**3.42**

TABLE 6.6: Accuracy comparison after adopting MixUp, Label Smoothing, Cosine Scheduling.

Mixup_Label_Multistep	**ResNet50**	**ResNet34**	**ResNet18**
HDL	36.58	36.05	36.32
BL	38.42	37.63	37.37
Change	-1.84	-1.58	-1.05
Stem	63.13	61.99	60.2
BL	61.01	61.34	58.73
Change	**2.12**	**0.65**	**1.47**
Fruit	68.14	66.03	63.65
BL	65.28	62.02	61.86
Change	**2.86**	**4**	**1.79**
Flower	79.07	78.52	77.01
BL	79.29	74.87	74.39
Change	-0.22	**3.65**	2.62
Leaf	71.65	-	-
BL	65.28	62.02	61.86
Change	**6.37**	-	-

TABLE 6.7: Accuracy comparison after adopting Mixup, Label Smoothing and MultiStep Scheduling.

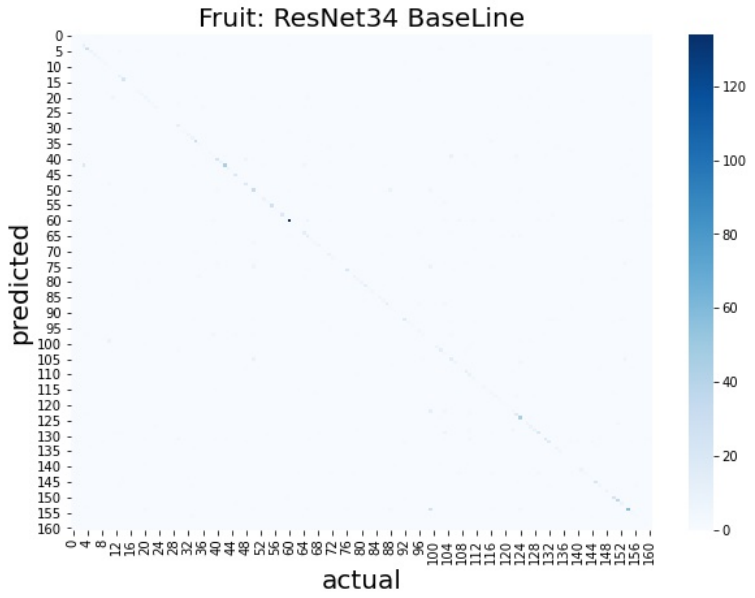


FIGURE 6.1: Confusion matrix for the fruit organ:Baseline ResNet34.

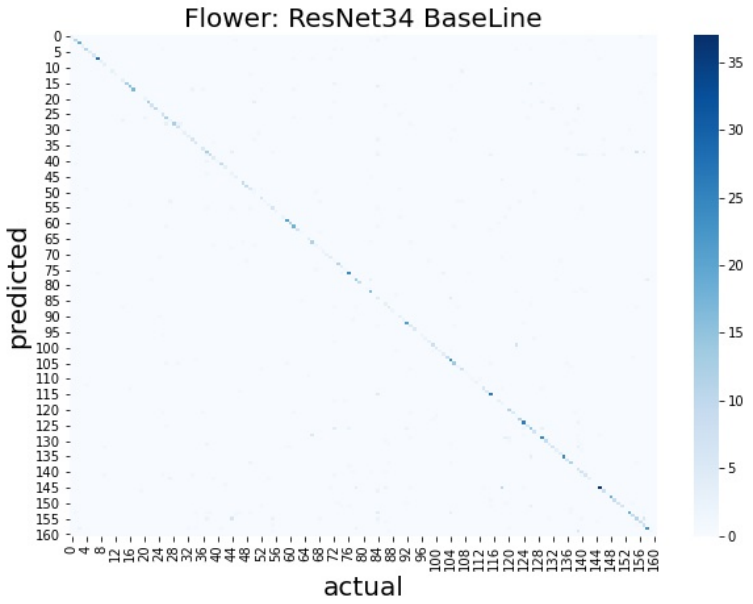


FIGURE 6.2: Confusion matrix for the flower organ:Baseline ResNet34.

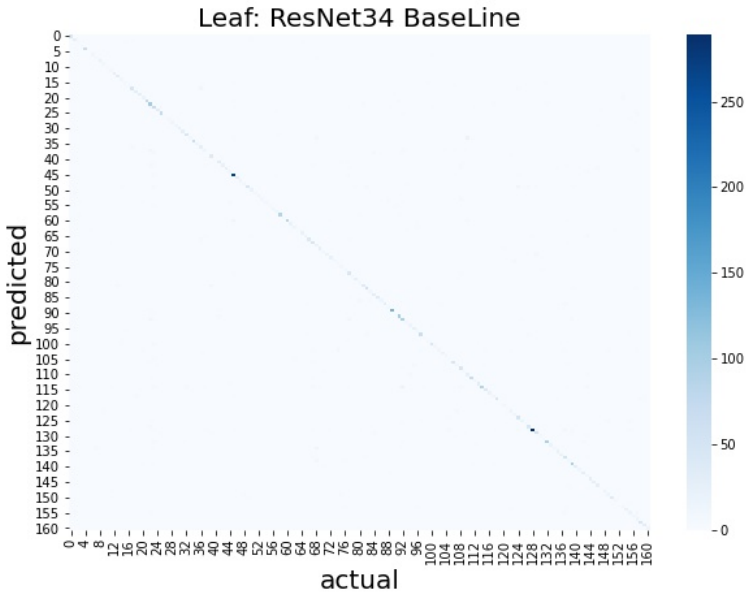


FIGURE 6.3: Confusion matrix for the leaf organ:Baseline ResNet34.

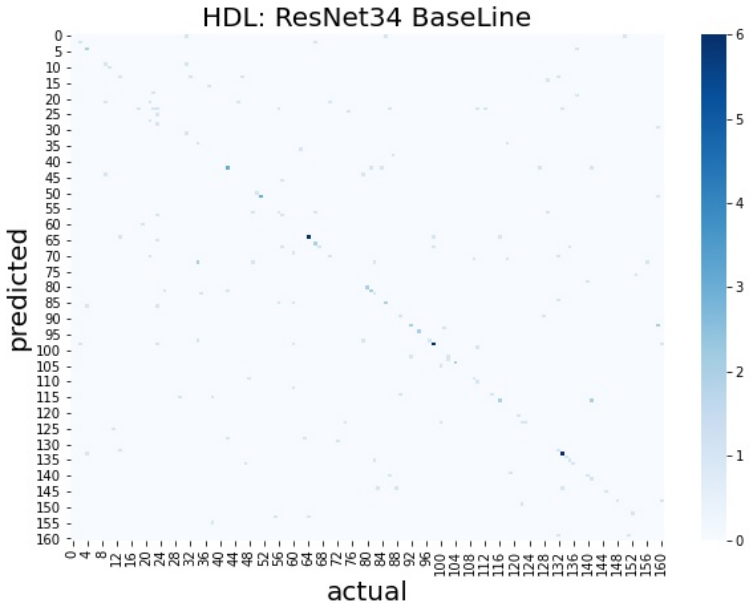


FIGURE 6.4: Confusion matrix for the HDL organ:Baseline ResNet34.

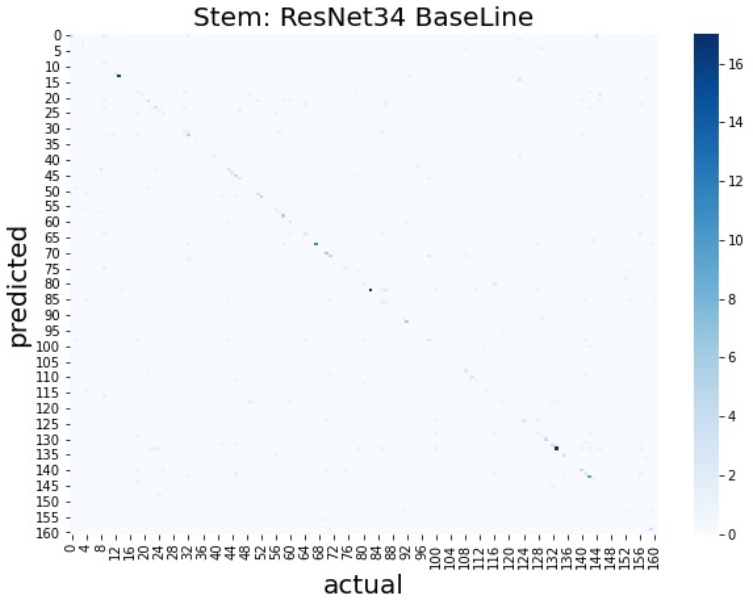


FIGURE 6.5: Confusion matrix for the stem organ:Baseline ResNet34.

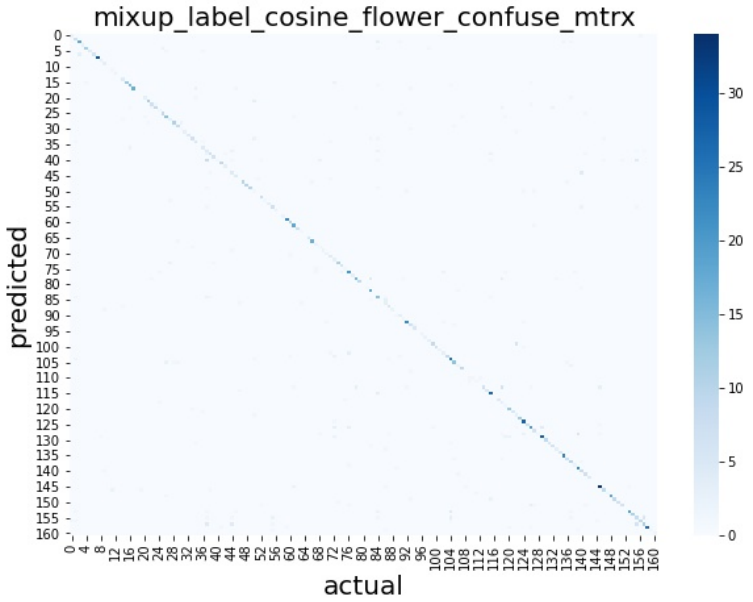


FIGURE 6.6: Confusion matrix for flower organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Cosine Scheduling.

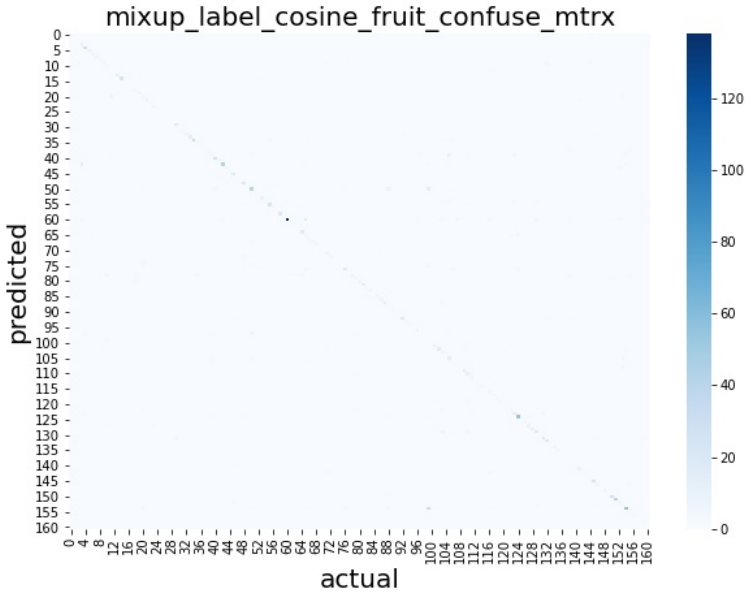


FIGURE 6.7: Confusion matrix for fruit organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Cosine Scheduling.

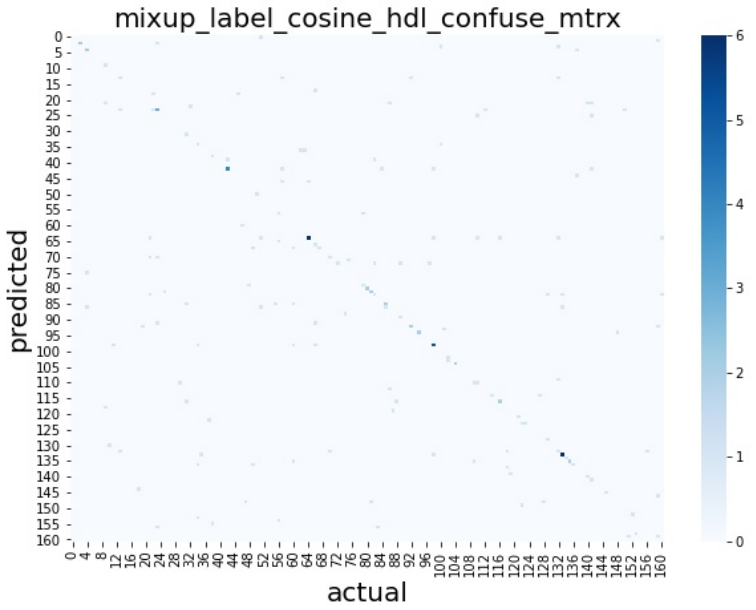


FIGURE 6.8: Confusion matrix for HDL organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Cosine Scheduling.

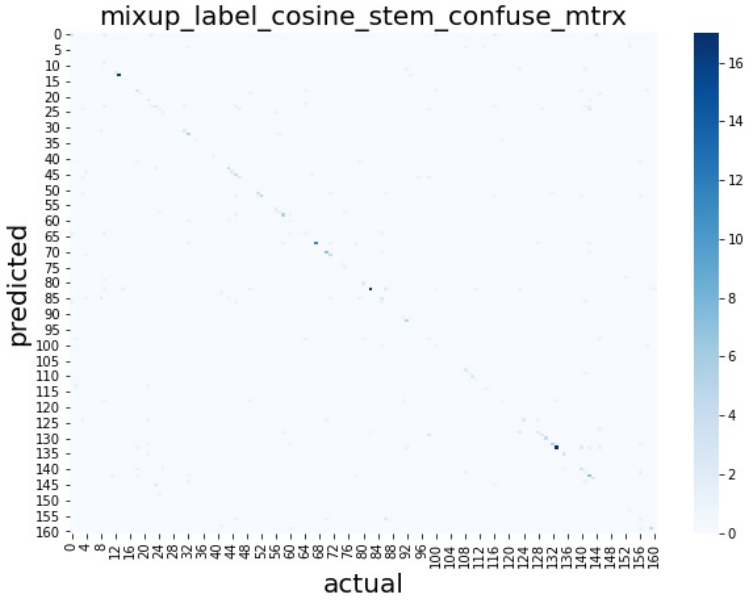


FIGURE 6.9: Confusion matrix for stem organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Cosine Scheduling.

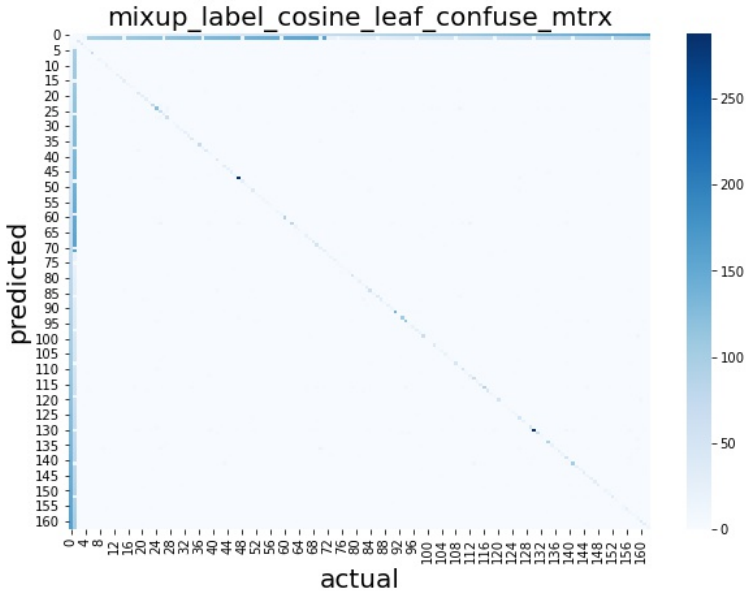


FIGURE 6.10: Confusion matrix for leaf organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Cosine Scheduling.

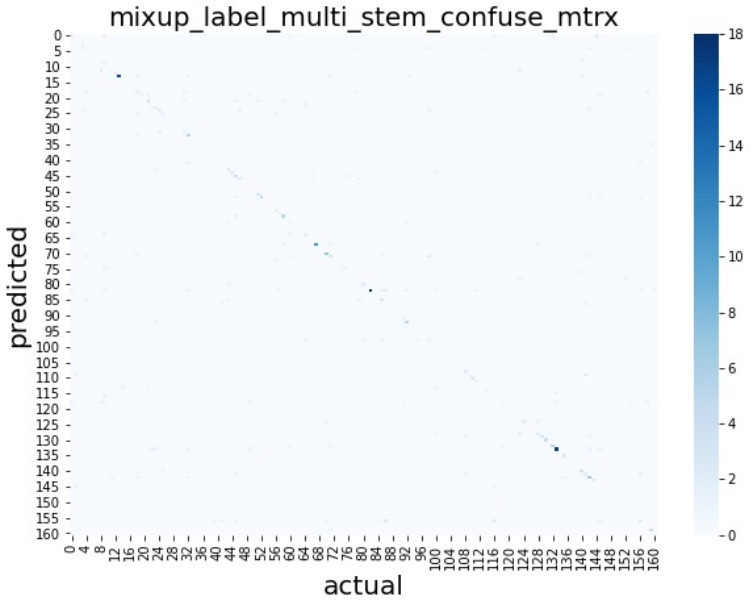


FIGURE 6.11: Confusion matrix for stem organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Multistep Scheduling.

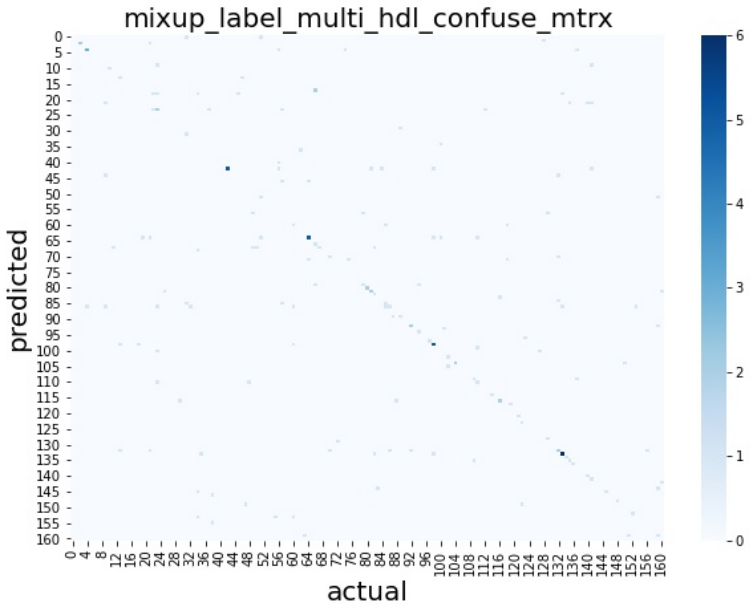


FIGURE 6.12: Confusion matrix for HDL organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Multistep Scheduling.

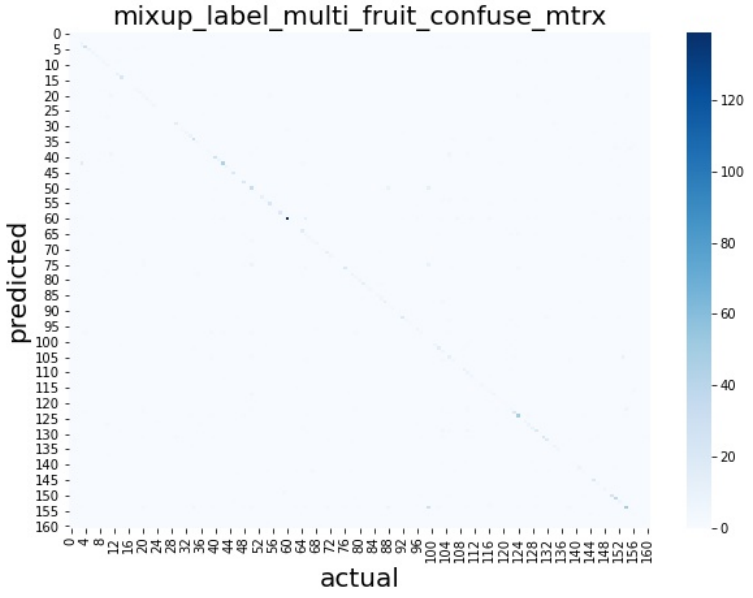


FIGURE 6.13: Confusion matrix for fruit organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Multistep Scheduling.

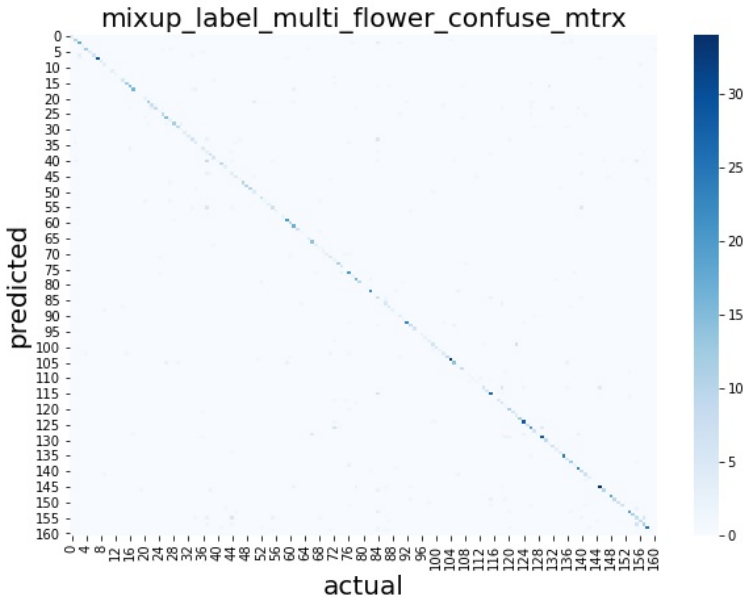


FIGURE 6.14: Confusion matrix for flower organ:
a) Architecture: ResNet34.
b) Optimizations: MixUp, Label Smoothing, Multistep Scheduling.

Chapter 7

Conclusions

7.1 Future Work

Although performing the mentioned techniques can boost up the training or improve the accuracy, they could not resolve the variability in the dataset. Though they belong to different species, most of the respective organ specific images look alike and thus we could not get the best accuracies. This is also because of the long-tailed distribution of Fine-Grained Visuals in the wild as shown in Figure 7.1 In other words, due to imbalanced distribution of organ specific species in the data, it is necessary to treat all classes in the dataset to have an equal distribution of data either by creating more data from existing dataset by performing different augmentation techniques or increase the sample size by synthetically increasing the samples such as implementing on the fly image augmentation or by SMOTE technique. Many feel that performing synthetic augmentation will increase the load on the GPU. But some researchers suggested that this technique works better by using proper resources like performing training on the GPU and performing synthetic augmentation on CPU to achieve good results.
In order to increase the accuracy, it is also important to improve the quality and quantity of the dataset. Performing techniques like undersampling and oversampling simultaneously can perfectly balance the dataset.
Recently, our team has worked to treat the imbalances in the dataset by enhancing the dataloader which does data augmentations, data transformations on-the-fly. This approach does oversampling on the entire dataset, creating equal number of samples in all the respective organ specific classes. During this approach, though we have treated the imbalanced data, we were unable to achieve the best

Organ	Count	Imbalance Treatment
leaf	130508	Undersampling and Oversampling the data
flower	88942	Undersampling and Oversampling the data
fruit	32714	Data Augemtations and Randomsampling
bark	4625	Oversampling data byAugmentation techniques
HDL	5738	Oversampling data by Augmentation techniques

TABLE 7.1: Imbalance treatment suggestions.

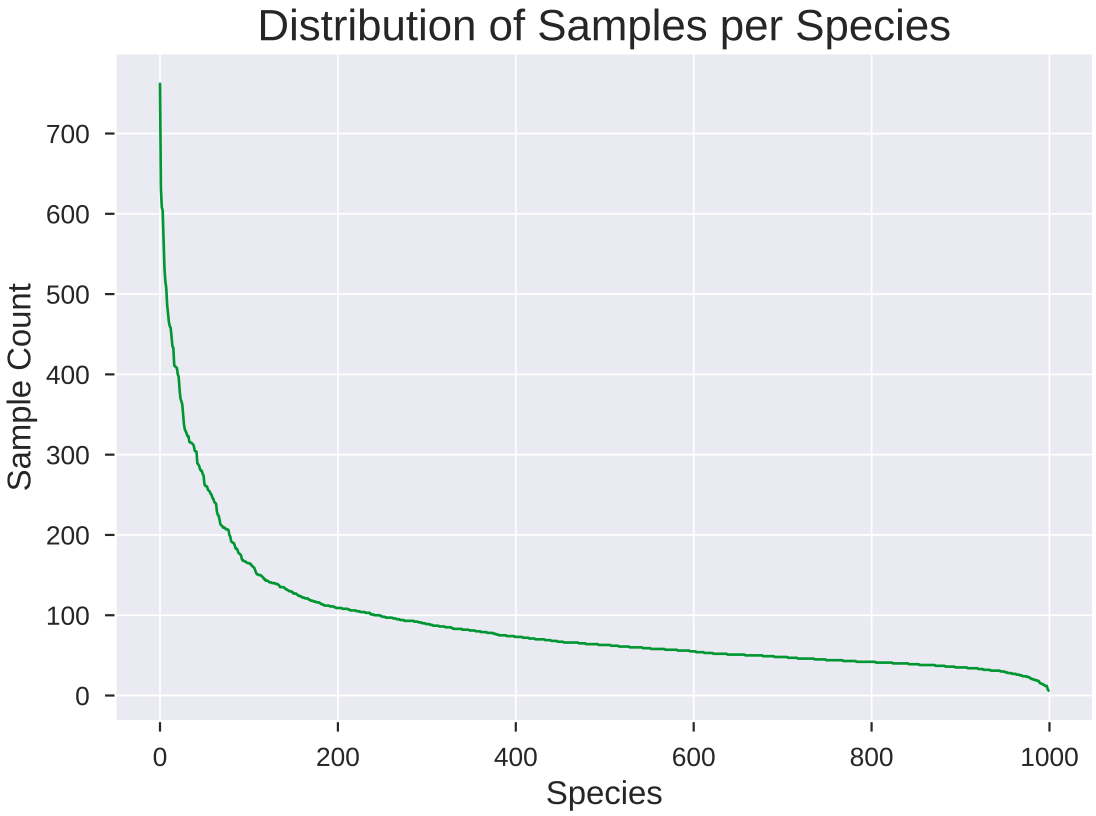


FIGURE 7.1: Long-tailed distribution of DARMA dataset species.

results as the model converged too quickly during the early stages of training and also as there are a lot of similar data. If this multi-class imbalance problem is resolved by implementing the suggested techniques from the Table 7.1, we can achieve better results and can obtain better plant organ models which can be used to train many other fine-grained plant visualization classifiers and can make use of them in multiple applications.

7.2 Conclusions

In this report, we introduced an approach of organ based plant species identification in the wild by developing organ species specific classifier with ResNet as backbone. In order to perform this fine-grained visual classification in the wild, we have created a custom dataset with over 2 million fine-grained visuals of plant organ species and coined the dataset as DARMA. DARMA is one of its kind as there is no such dataset with high quantity and quality with 1000 species of plant organs in the wild. DARMA stands for Detection As Reinforced Means of Attention, since we have gathered the dataset by creating bounding boxes, around the organ species and thus using the annotations for the purpose of our detection. By fine-tuning the dataset and performing some training refinements such as optimizing the model by introducing Learning Rate scheduler, label smoothing and synthetic data-augmentation such as MixUp and also learned that implementing distillation techniques can improve the performance of the model. Thus, unlike many conventional datasets, we can make use of DARMA in multi real-world applications. Since dataset distribution is long tailed, Along with optimization techniques, treating the imbalance data in the respective organ species class can achieve great accuracy with reduced error.

References

[1] H. Zheng, J. Fu, T. Mei, and J. Luo, "Learning multi-attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5209–5217.

[2] M. Sun, Y. Yuan, F. Zhou, and E. Ding, *Multi-attention multi-class constraint for fine-grained image recognition*, 2018. arXiv: 1806.05372 [cs.CV].

[3] W. Luo, X. Yang, X. Mo, *et al.*, *Cross-x learning for fine-grained visual categorization*, 2019. arXiv: 1909.04412 [cs.CV].

[4] G. V. Horn and P. Perona, *The devil is in the tails: Fine-grained classification in the wild*, 2017. arXiv: 1709.01450 [cs.CV].

[5] M. Šulc and J. Matas, "Fine-grained recognition of plants from images," *Plant Methods*, vol. 13, no. 1, pp. 1–14, 2017.

[6] A. D. Chapman *et al.*, "Numbers of living species in australia and the world," 2009.

[7] H. Goëau, P. Bonnet, and A. Joly, "Overview of lifeclef plant identification task 2020," in *CLEF 2020-Conference and labs of the Evaluation Forum*, 2020.

[8] H. Göeau, A. Joly, and B. Pierre, "Lifeclef plant identification task 2015," *CLEF working notes*, vol. 2015, 2015.

[9] M. R. Keaton, R. J. Zaveri, M. Kovur, C. Henderson, D. A. Adjeroh, and G. Doretto, *Fine-grained visual classification of plant species in the wild: Object detection as a reinforced means of attention*, 2021. arXiv: 2106.02141 [cs.CV].

[10] S. pidhorskyi, *Annotator*, https://github.com/podgorskiy/anntoolkit, May 2020.

[11] MatthewKeaton, *Snappy annotator*, https://github.com/wvuvl/snappy_annotator, May 2020.

[12] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[13] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, *Bag of tricks for image classification with convolutional neural networks*, 2018. arXiv: 1812.01187 [cs.CV].

[14] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, *Mixup: Beyond empirical risk minimization*, 2018. arXiv: 1710.09412 [cs.LG].

[15] R. Müller, S. Kornblith, and G. Hinton, *When does label smoothing help?* 2020. arXiv: 1906.02629 [cs.LG].

[16] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[17] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, "Revisiting knowledge distillation via label smoothing regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[18] S. Thulasidasan, G. Chennupati, J. Bilmes, T. Bhattacharya, and S. Michalak, *On mixup training: Improved calibration and predictive uncertainty for deep neural networks*, 2020. arXiv: 1905.11001 [stat.ML].