**Project Overview**

As University of Pretoria stated, a successful cultivation of crops broadly depends on the weed control strategies. It has been widely observed that tons of cultivated crops are wasted due to crop infestations. A zero percent loss in crops is entirely a rare event, generally there is 10 to 100% losses as per the current weed control practices. (Pretoria, n.d.)

Additionally, for farmers it is of vital importance to detect weed in the initial first to six weeks of plantation, because both weed and crop vigorously search of nutrients and water in the soil for this specific period.

According to the research paper published by Thomas Mosgaard Giselsson and co, there is no robust computer vision system out there, which can classify the species of ground-based weed. (Giselsson, Jørgensen, Jensen, Dyrmann, & Midtiby, 2017)

In this project, I have developed a CNN model which can detect plant species up to 92% of accuracy.

**Problem Statement**

Here, I am using a dataset which contains 5,539 images of crop and weed seedlings. The images are grouped in 12 classes. Moreover, these classes represent common plant species from Danish Agriculture. (Dyrmann)

Each classes contain RGB images, which reflect various stages of plant growth.

Using this dataset, the goal is to build a model which can further classify weed seedlings and crops.

Moreover, I have also planned to further extend this project by integrating this Deep Learning API with a web or mobile application, where a farmer will just have to upload the image in the mobile or web application, and the API will be able to predict the species of the weed or crop, thus, farmer can take appropriate action. In this way, if a farmer found a specific seedling as weed, it can be destroyed before it infestate the actual crop.

The actual data set can be found out from the link mentioned below.

<https://vision.eng.au.dk/plant-seedlings-dataset/>

**Metrics**

The development of machine learning models revolves around the idea of constructive feedback principle. When we build a model, the evaluation metrics are used to get feedbacks, and thus further improvement is continued to achieve a desired accuracy. (Srivastava, 2016)

Simply building a predictive model is not the prime motive, we must utilize the capability of evaluation metrices to discriminate among model results.

In this project, the evaluation metrices that I have used to validate the model, and for further improvement of model accuracy are Validation Accuracy, Precision, Recall, F-1 score, and Confusion Matrix.

* Validation Accuracy is where both true positive and true negative is taking into consideration.

Validation Accuracy =

* Precision: It tells the accuracy of positive predictions.

*Precision =*

* Recall defines the fraction of positive cases that the model is able to catch or correctly identified.

*Recall =*

* F1 scores tells the percent of correct positive predictions. It is nothing but the weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst score is 0.

*F1 Score =*

**ANALYSIS**

**Data Exploration**

The dataset consists of around 5,539 images and weed seedlings. All the images are further classified into 12 groups or classes.

These classes are nothing, but the common plant species from the Danish Agriculture.

Image Types: All the classes contain RGB images in .png format having different sizes.

Below is a list of all the 12 classes:

A screenshot of a computer

Description automatically generated with low confidence

Moreover, as an initial data exploration step, I want to investigate the count of images present in each class such that I can examine whether the dataset is balanced or not.

Text

Description automatically generated

Image shape is in 3 dimensions. Below is the shape of a randomly chosen image.



**Data Visualization**

First and foremost, I want to plot images in a clean, tight grid like structure, along with its class name(species). The approach which I adopted was to first create a pandas data frame with filename and target.

*Note: Here, filename denotes the path of the image, and target is the species name.*

Using this strategy, I would set up the data for further preprocessing, where it will be further balanced, and split into train and test sets.

Table

Description automatically generated

***Fig1. Denotes a pandas dataframe with filename and target as label.***

Finally, I plot the images in a grid format having 3 rows and 5 columns, where I labeled each images with its species name.

Graphical user interface, application

Description automatically generated

***Fig2 is a plot of 25 images having its species as its label.***

For Data balancing, I first performed some exploratory data analysis and visualization to fetch information about the overall data distribution.

Chart, bar chart

Description automatically generated

***Fig3 is a bar plot, which display the information regarding the counts of all images categorized based on species type.***

From the above figure, it is quite evident that the distribution of data is quite contrasting in nature, where Loose Silly-bent is having 762 images as compared to common wheat, which is only 253.

This specific situation will be highly problematic in a classification scenario, as it will leads to misleading accuracy of classes. To solve this issue, I have equally distributed the images for each classes. Because of this, now each class will have necessary amount of information for training a robust, and accurate model.

Also, another important aspect about neural network is, that while building the architecture of CNN, we have to decide the layers of CNN based on the input size of images. Also, the images should have same size while feeding to the input layer.

Thus, it is always good to resize the images, especially to smaller size before feeding to CNN.

Since, we have to investigate the size of images, therefore I performed a jointplot of seaborn library, which basically plots the dimensions (height & width) of all the images in x and y axis.

Chart, scatter chart

Description automatically generated

***Fig4 gives information about the shape of the image ranging from 0-500 to >3500, therefore in preprocessing steps we will resize the images.***

**Methodology**

**Data Preprocessing**

The steps which are involved in the data processing are mentioned below:

1. First, I copied all the images into a file, and then according to each class(species), I wrote a logic which balances the entire dataset.

Graphical user interface

Description automatically generated with low confidence

***Fig6: This is how the dataset looks like after balancing.***

1. Now, I created a directory named which contain two subdirectory’s called train and valid.
2. Using the scikit learn train test split method I split the dataset (Fig6) into train and validation.
3. After splitting, I reshaped all the image were I assigned equal width and height to all of them, and using opencv imwrite method, I copied all the images into their respective train and test subfolder. Here, the main Idea is to allow ImageDataGenerator to load the images from these subdirectories. Thus, I have followed this specific format to save image files.

Text

Description automatically generated A picture containing text, black

Description automatically generated

***Fig7: The figure display the count of images in train and validation folder.***