Flower Power III

Large-scale Flower Monitoring

Final Report

September 2021 - January 2022

FR Corp.
Veneta Angelova
Lia Boyadzhieva
Kristina Krasteva
Filip Vangelov

Summary

Flower Power III is a project launched by the Fontys Research Group “AI & Big Data” and it is carried out for a third year in a row, hence the name. Their representative and our client Gerard Schouten approached us with a request to come up with a feasible and optimal solution for the monitoring and recognizing of wild flowers in the Netherlands. This is a complex and important issue nowadays which must be addressed as soon as possible. The drastic decline in the biodiversity can lead to the so-called multiplier effect which can affect more species and cause other ecosystem related disturbances.

What is requested from us is a solution assisting with the monitoring and recognition of different wild flowers via the help of a deep learning AI model. To perform the project, we were given access to the Fontys Research Drive where 11 GB of high quality flower photos were stored along with an Excel file providing the details.

For the classification problem that we were faced with, labeled data was needed. However, ours was not and thus we had to invent an automated solution that would not take months to execute. Such a piece of code was written by us which uses HSV as a color detection method with a trackbar slider for an easy filtering of the colors. Together with it, a cropping technique was integrated and in this way we were able to prepare our data properly.

We applied three modeling techniques by the end of the project - Faster R-CNN, a custom model crafted by us and the pre-trained model MobileNet_V2. All of them have concrete specifications that need to be carefully taken into account and therefore they are extensively explained in this report. The modeling techniques ended up with impressive accuracy results through which we managed to achieve our project goal.

During the ADS Minor, our group was fully devoted to making this project happen in the best way possible and both we as a team and our mentors and client were extremely satisfied with the obtained results and the delivered end product.

Table of Contents

Introduction

With the rapid development of technology and urbanization over the last 50 years, the biological cycle has profoundly changed. Farming has become a leading industry in many countries, the Netherlands including. The excessive use of land for profit and the polluting of the soil with pesticides has undoubtedly influenced the biodiversity of the natural landscape. Wild flowers are part of the ecosystem that this project aims to focus on.

Large-scale flower monitoring and counting is still extremely challenging since it is arduous, requiring a lot of labor force and prone to errors as it is performed manually. By coming up with an AI model to monitor, recognize and count the flower species, our group hopes to raise more awareness to what species might be endangered and to prevent the loss of biodiversity. Furthermore, such an AI model would also provide systematic surveys which would contribute to the discovery of trends in flower occurrence over time.

This document consists of 8 chapters in which our team will describe all the necessary information regarding Flower Power, the process of execution of this project, and give recommendations to the client or future groups working on the project. The content of each chapter is outlined in the section below.

Chapter overview:

- **Chapter 1** introduces the company and the client along with the current situation, the business requirements and goals so that every reader gets an overview of the project.
- **Chapter 2** gives an overview of the data - the amount, the sources, how it was retrieved and formatted in order to be used for the machine learning experiments.
- **Chapter 3** describes in detail the process which is based on the CRISP-DM model taking into consideration the 6 phases - Business understanding, Data understanding, Data preparation, Modeling, Evaluation and Deployment.
- **Chapter 4** focuses on the results which are achieved throughout the project.
- **Chapter 5** provides the readers with a discussion regarding the ethical point of view of the Flower Power III project.
- **Chapter 6** wraps up and summarizes the findings and results from the executed experiments.
- **Chapter 7** intends to provide the client with key recommendations based on what results and research our group has obtained.
- **Chapter 8** includes critical reflection of the work performed by the team during the project.

1. Business Understanding

This chapter provides the reader with an explanation why the project should be initiated and what the expected value of the results will be. It elaborates on the business requirements and why commencing the project is expected to result in a solution that addresses these requirements.

Currently, we are aware that the main goal of our client is to monitor flower species, which includes the number and the type of species in a particular area. However, the idea behind this goal is, by taking this initial "step" of initiating this project, to develop into a much larger scale and help to work towards achieving the United Nations's goals. Furthermore, as pointed out by our client, the reduction of the number of wild flowers has a multiplier cascading effect on other species as well, such as bees and insects, which on its own affect mammals and birds, and all of those species affect us and our ecosystem. This ecosystem disturbances also have great business value for our client as well.

However, in order to track the effects on each of the species in the above-mentioned ecosystem our team will start at the bottom of the web-of-live by creating a model which is able to differentiate between different species and count them. We will also use the geographical data provided in order to plot our findings into interactive visualizations. Moreover, our solution will be "sustainable" in a way that if the client has more or different data he can still use our model without any major adjustments.

1.1 The company

The formal client for this project is Gerard Schouten, professor AI & Big Data at Fontys University of Applied Sciences and head of the lectorate AI & Big Data. Together with Naturalis Biodiversity Center, they have formed the Flower Power research group. Their aim is to understand flower biodiversity better in a sustainable way.

1.2 Current situation

Together with Naturalis Biodiversity Center, our client has gathered a large data set of the flower species in and around Eindhoven. Furthermore, they have also partially labeled the data they have collected, however currently they do not have the technology to turn this data into valuable insights. Additionally, it is important to mention that the data (pictures of flower species) was collected using a DSLR camera. This is important to be mentioned because the idea of our client's research group is, if possible, to collect such type of data in the future by using drones, which would bring large scale monitoring of species on a whole new scale.

1.3 Business goal

The goal of the project is to monitor and differentiate between different types of flower species over large areas, which would have a great impact on biodiversity and ecosystem related complications. Provided that our project is successful, our model would give valuable insights such as areas with tremendously decreased amounts of wild flowers, insights regarding the multiplier effect on other species affected and other ecosystem related disturbances.

1.4 Business requirements

In this section you can find a more explicit explanation of the requirements gathered during the interviews with the formal clients that the development team should deliver as an end product. We have separated the requirement in two groups, Mandatory , which is the first priority and highly required from the clients, and Additional, which might be added from the group to the overall performance of the model with correlation to the overall business goal of the project.

"Must-haves" Functional Requirements

Starting with what the final product should deliver, we are going to develop a supervised Machine Learning model using deep learning techniques specifically used for image classification and recognition in order to create this new algorithm that can detect, identify and count different species of flowers. This model should be able to:

- Identify different species of flowers based on the provided testing data from the company, this data shows images from around 2.5 meters of height (in a bird view).
- Count the amount of different and same types of species on the given testing picture.
- Store the output information such as number of flowers, type, etc. in a convenient format.
- Provide some data visualizations of the output data (results).

"Could-haves" Requirements

Here is the list of the proposed additional requirements that were discussed during the initial company/client - team meeting, these additions are "nice to have" and will provide potentially better results and a more efficient model:

- Instead of using the provided data sets from 2.5 meters high, use drone pictures found online that can be used to feed the model and classify images from a drone view, since this is going to be the overall goal of the company.
- Create a labeling system which can provide metadata for the images (.json or .xml)

2. Data Understanding

In this part of the document detailed information about the data that our team will work with is presented.

2.1 Data sources

The Eindhoven Flower Power dataset contains 817 high resolution photos of different flower species. These images were taken from April to July, 2021 around Eindhoven, the Netherlands in various landscape types. All images were photographed from a top view angle (2-2.5 meters) with a tripod. On some of them there are multiple flower species per image. No labels are assigned to the images. This dataset can be found on the Fontys Research Drive and hence currently, it is not open-source.

2.2 Amount of data

The raw data from the Fontys Research Drive consists of the following:

- Three main folders (April2021, May2021, June2021, July2021)
    - In April2021 there are 9 sub-folders of the locations visited in April
    - In May 2021 there are 20 sub-folders of the locations visited in May
    - In June2021 there are 13 sub-folders of the locations visited in June
    - In July2021 there are 17 sub-folders of the locations visited in July

- In total, the data is 11.5GB and each photo is 6720x4480 pixels. Per image the GPS coordinates are known (latitude & longitude), except for the month of April.
- The format per photo is: <_photo_id>.JPG

In addition to the photos, our client has provided us with an excel sheet where the name of the flower for each image is present along with information about the location, date, etc. Below you can find the full description of this spreadsheet.

<u>The landscape type can be:</u>

- Urban area (park, public area, industrial park)
- Roadside, dike, railway border, wasteland
- Extensively farmed cropland
- Dunes
- Grassland, wet meadow
- Fen meadow, marsh
- Pool, ditch, fen

<u>The unit can be:</u>

- Single flower
- Spike
- Head
- Raceme
- Umbrel
- Cyme
- Corymb

Eindhoven Flower Dataset 2021	806 rows
Column	**Description**
Close-up	If the image was taken from a small distance
Human element	If there is another object apart from the flower on the image
Visit	A number indicating on which visit the image was taken
Date	The date when the image was taken (+ what type of visit it was, i.e., walk, bike ride)
Location*	Where the image was taken (+ the landscape type)
Photo ID	The image's unique identificator
Main species	The name of the flower species (in Dutch)
Latin name	The latin name of the flower species
English name	The english name of the flower species
Status identification	Check if the flower is the correct species
Count	The number of the flowers per image
Unit*	What kind of a flower is in the image (i.e., single flower, spike, etc.)
Other visible flowers	What other flower species can be seen per image
Remarks	Additional comments of the client

Table 1. Eindhoven Flower Dataset 2021 xlsx sheet

3. Process

For the Flower Power III project the standard IBM CRISP-DM model was used. There are 6 phases to be considered – **Business understanding, Data understanding, Data preparation, Modeling, Evaluation and Deployment**. These stages are linked to one another to help us solve the particular data science problem.

Below a more in-depth description of what are the major tasks per stage is shown.

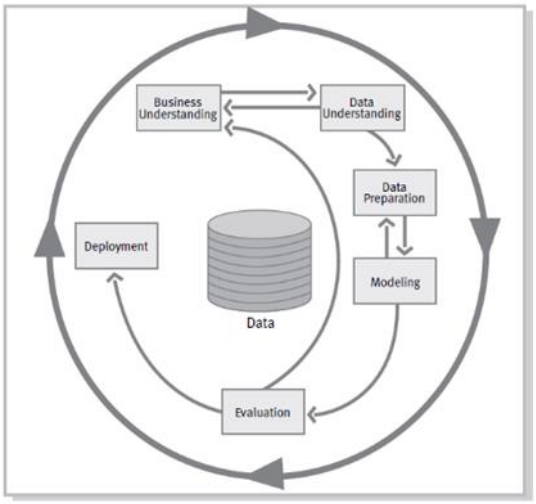

Figure 1. CRISP-DM cycle

3.1 Business Understanding

Before implementing any solution, the problem needs to be understood and the goal to be determined. This step is of great importance as it will result in a more reliable starting point for the actual modeling.

The duration for this phase was 3 weeks in which the following tasks were completed:

3.1.1 Business Proposal

- **Initial meeting with the client and the technical tutor** - requirements gathering, explanation of the project in more details
- **Determine the business objectives** - background (what is the influence and impact of monitoring wildflowers), scope, requirements and planning

3.1.2 Assess the situation

- **Inventory of resources**

In terms of personnel, our team has a total of 4 data analysts at disposal for the project. Weekly there were 2 set days (Tuesday and Wednesday) in which 3 members worked on the project. These 3 analysts accounted for 48 project hours weekly. Depending on the workload of our group, the capacity increased, however never decreased (exceptions; being a member falling out, fallen ill, etc.), in case of an exception, arrangements within our mini company (FR Corp.) were made to either have a company member stand-in, or to reschedule the missed hours. Our

team had a fifth member occasionally working on Friday which also served as part of his Open Program.

Data sources were provided to the team members by the client in the Fontys Research Drive environment. The dataset is unique in the sense that it covers the entire flower season, all images are collected in the wild. For instance, there is a lot of background clutter and sometimes the flowers are even partly occluded (i.e., the images do not contain clear close-ups of flowers). Furthermore, many images contain a mixture of flowers (i.e., one image might contain 5 flower species and in total 107 individual flowers). Additionally, all images are with high resolution.

For technical facilities, the group used their own laptops, accessed at their home networks, as well as the Fontys FHICT. Computations were done on the local machines of the personnel mainly by using Python and Jupyter Notebook.

- **Constraints**

The purpose of the below mentioned constraints is to outline what restricts or dictates the actions of this project. There are some general constraints that can occur in almost every project, but we have also included some more specific ones which are tightly related to Flower Power III.

Constraint table		
ID	**Constraint description**	**Explanation**
C1	Time constraint	The project must be finished within 18 weeks. It starts on September 6, 2021, and is expected to be completed on January 21, 2022.
C2	Dataset constraint	The image dataset provided does not have labels assigned per photo.
C3	Resource constraint	Not all code can be easily run on a standard PC and requires more computer power
C4	Scope constraint	The project focuses on photos taken from 2.5 height and not taken by drones
C5	Programming language constraint	Since all members are familiar with Python, the code will be written in it (instead of R or Matlab)

Table 2. Constraints

- **Risks**

Every project has potential risks that can affect to normal development of the product, in this table we will present the most related risks for a project of this type, the effect that they can cause in terms of damage to the final product and development, the probability of such risk to happen and the mitigation method or how to resolve such risk. Effect and probability will be measured as Low, Medium and High.

Risk assessment				
ID	**Risks description**	**Effect**	**Probability**	**Mitigation**
R1	The image recognition model does not reach the wanted level of accuracy	High	Medium	Re-evaluate the model and spend more time on training data, try different machine learning model
R2	The image recognition model does not reach the wanted level of efficiency	High	Medium	Re-evaluate the model and prepare the training data, try different machine learning model
R3	Not enough images for the training set	High	Low	Request more images from the client, research online for more data, but also validate with the client
R4	Bad image quality is bad, which causes troubles with identification/counting of different flowers	High	Medium	Configure the effects of the pictures and try to make the flower part more visible for the model
R5	Running and training the model takes too much time and it causes troubles with the group work processes	Low	Medium	Use the supercomputer in TQ - 5
R6	A member leaves the group	Medium	Low	Reschedule and spread the work among the remaining members, asking for help from other groups.

Table 3. Risks

3.2 Data Understanding

When the business part of the process was clarified, we proceeded with the data understanding stage. It was an extremely essential phase since it provided us with first insights about the data.

The duration for this phase was 2 weeks in which the following tasks were completed:

3.2.1 Collecting initial data

- **Accessing and gathering the data**

Our group was given access to the dataset on the Fontys Research Drive. Initially it consisted of 3 folders for the months of April, May and June and eventually the data was enriched with photos from July. In total, there were 814 high quality images. The dataset was downloaded by each team member.

3.2.2 Describe data

We got our first impression on the dataset by describing it in detail for our business proposal document. Each of us got familiar with the folders, the subfolders and the excel sheet provided to us by the client. For more information about those two please refer to the second chapter of this document.

3.2.3 Explore data

- **Exploratory Data Analysis (EDA)**

To understand the dataset better, our group has prepared an EDA Jupyter notebook with the following techniques being applied in it:

a) *Pandas profiling* which is an open source Python module for quickly doing an EDA with just a few lines of code. The report generated a lot of valuable information regarding the data - missing values, distinct values, value count, etc. It also displayed the first and last 10 rows of the dataset.

b) *Unique flower species* - we worked with the english names of the flower types and we were curious to see how many of them there were in the excel spreadsheet. As you can see below, a flower species is counted twice if there was a whitespace present after the name.

```
1  df['english_name'].unique()
```

```
array(['Wood Anemone', 'Red Dead-nettle ', 'Red Dead-nettle',
       'Lesser Celandine', 'Common Dandelion', 'Lesser Celandine\xa0',
       'Daisy', 'Chickweed', 'Marsh-marigold ', 'Marsh-marigold',
```

Figure 2. EDA unique name

We simply removed the whitespaces and in this way instead of ending up with 129 “unique” flower types, we got 122.

c) *Visualizing the number of units* - for all of our graphs in this notebook seaborn was used. We plotted all types of units and noticed that three typing mistakes have been made. Therefore, we replaced the single values with the same name as the unit with multiple values. ("spike - Spike; "Umrel - Umbrel", “Single Flower - Single flower”).

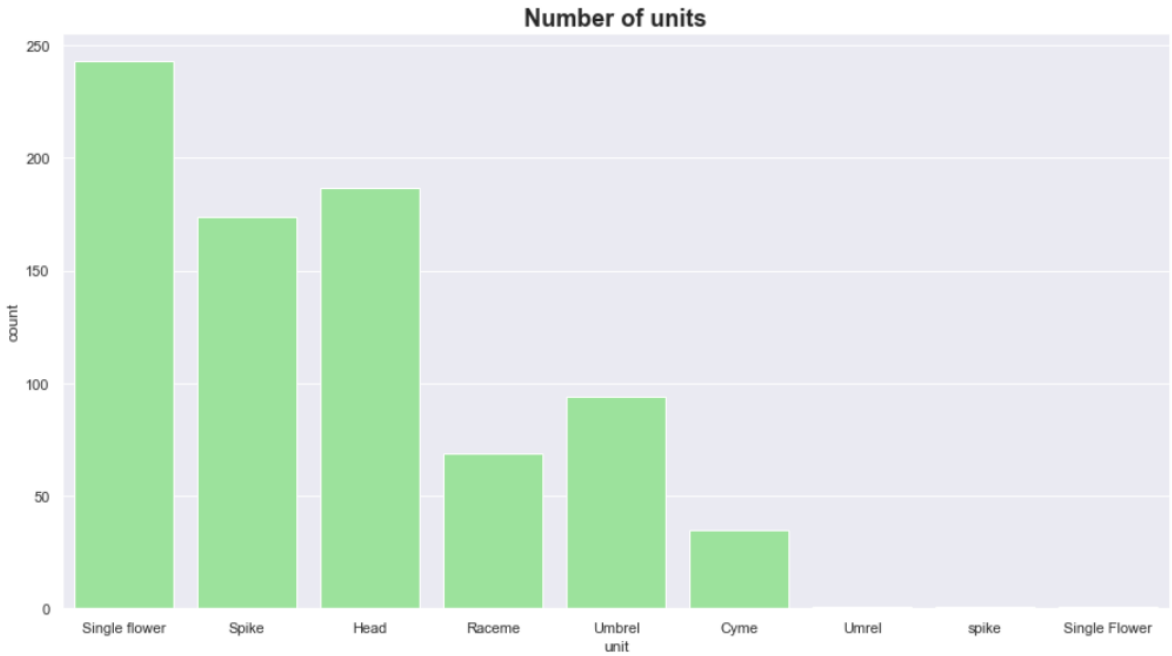

Figure 3. EDA units visualization

d) *Visualizing distribution of flowers per city and month* - There were a few instances where a location consists of multiple cities, as well as cities repeating. The reason for that might be because the images were taken in various districts of the cities. We discussed how to handle this data properly with the client since it can be used for further visualizations.

e) *Visualizing the 4 most common species in the dataset based on their unit type* - we created an interactive visualization in the form of a chord graph, by using the "Plotapi" library. The client can easily adjust our code and create new visualizations.

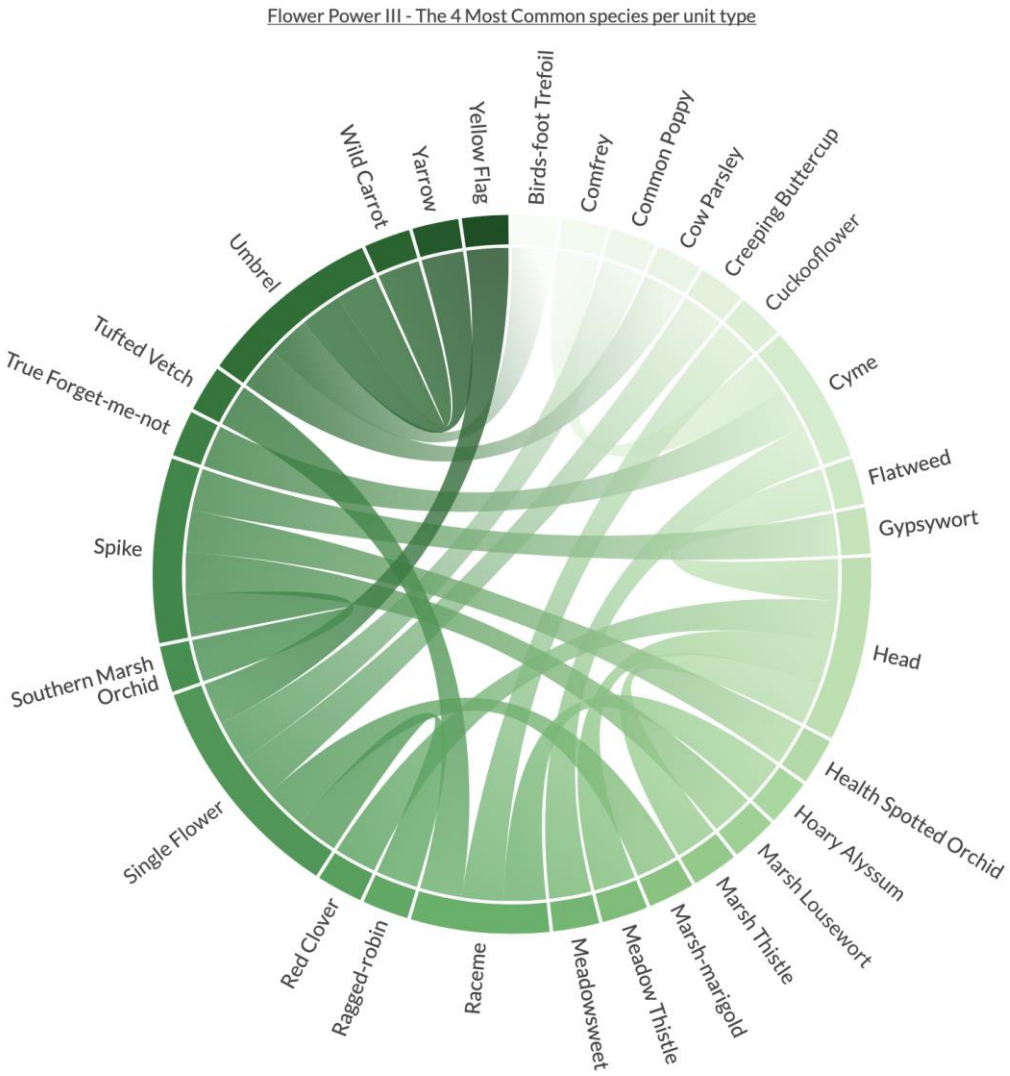

Figure 4. Chord graph

There are a few more visualizations in the full notebook which can be found on our GitHub.

3.2.4 Verify data quality

As a group we have concluded that there were a lot of discrepancies in the dataset provided. With this EDA, we were able to handle them in order to make more consistent and reliable data visualizations. From these graphs, we could observe that most of the pictures were taken in May and the bigger part of the flowers are of unit "single flower". However, with this data analysis, our group could not get many essential insights that would be beneficial for the future model.

3.3 Data Preparation

This stage was about preparing the final dataset that was fed into the model.

The duration for this phase was 4 weeks in which the following tasks were completed:

3.3.1 Select data

For the Flower Power project we decided to work only with the image dataset provided by our client and the excel file. As we would not have had the time to train a model for all 122 species, our client listed 15 to be the main focus for us. Eventually, we increased the number to 30 which were fed to the model.

3.3.2 Clean data

Since the spreadsheet was filled with a lot of information that was not relevant for the creation of the model and the analysis, we cleaned it both manually and via a Python code.

The alterations made by hand were the following:

- The tabs *Species Characteristic, Labelling Rules, Acquisition Settings, Student Questions, For Next Year, Effort Estimations, Notes on Inflorescence* were deleted so that the only sheet left was Flower Photos.
- From the Flower Photos tab the *Close-up, Human Element, Visit, Main Species, Latin name, Status identification, Count and Remarks* columns were deleted.
- All other columns were renamed so that they are all lower case.
- The heading *"Eindhoven Flower Dataset 2021 | All photos taken with Canon 5DM4, 24-105mm f/4 lens"* was removed.

3.3.3 Construct data

The alterations made in the Data Preparation Jupyter notebook are the following:

Since the column "location" contains the city and extra information regarding where the photo was taken such as district, we splitted the values of the column into two different columns - "location(the city)" and "district".

```
1  data['district'] = data['location'].str.split('|').str[1]
```

```
1  data['location'] = data['location'].str.split('|').str[0]
```

Figure 5. Data Preparation location

During this time, we started thinking of how we can label the data. Having the photos in subfolders based on location and month was not optimal and made working with the data very tedious and difficult. Therefore, we created a code to label the images according to the English name of the flower. We dedicated a whole week for this code, but it was worth it since it was later used in our modeling experiments. The logic behind is that it uses the photo id of the images in the folder and compares it with the photo id from the spreadsheet. Then, in front of the photo id of each image, the english name is added. Finally, we put all photos of the same flower species in a folder with the species' name so now instead of considering the month or location, the name is on focus.

Additionally, there is a function which concatenates the name of the flower to the name of the image. Also, the pictures which are not present in the excel file are stored in a different folder called "Extra images".

Furthermore, a csv file with the duplicated images from the dataset (2 photos having the same photo_id) is created. This code can be found in the Data Preparation Jupyter notebook as well.

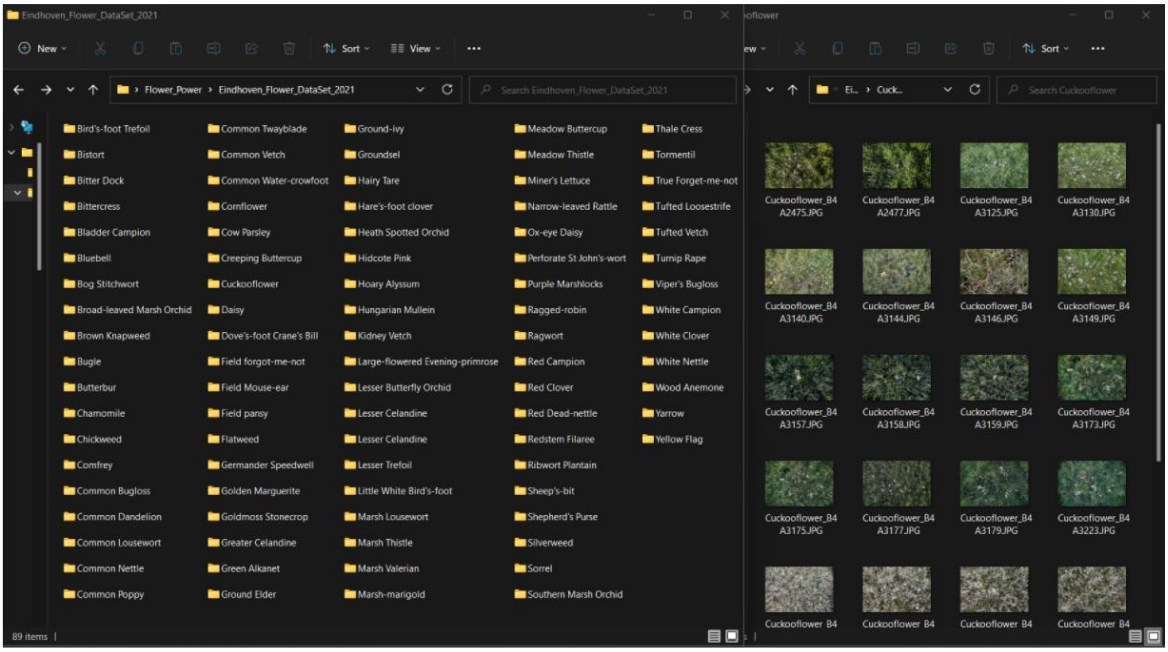

Figure 6. Data Preparation labeling

3.3.4 Integrate data

Since the date is only filled for the first row per visit, the following code fills in the NaN values of the rows with missing data. The function used is 'ffill' which stands for 'forward fill' and replaces the missing values with the corresponding value in the previous row.

From the date column we wanted to extract the month which was done by converting the date object into datetime format and then using the "strftime('%B')" function.

3.3.5 Format data

Finally, we synced the data when all of the above-mentioned techniques are applied. The rearrangement of the column positions led to the excel sheet looking like this:

	date	month	landscape	location	district	photo_id	english_name	unit	other_flower
0	2021-04-02	April	Woodland	Helmond	Warandabos	_B4A2382	Wood Anemone	Single flower	NaN
1	2021-04-02	April	Woodland	Helmond	Warandabos	_B4A2389	Wood Anemone	Single flower	NaN
2	2021-04-02	April	Woodland	Helmond	Warandabos	_B4A2399	Wood Anemone	Single flower	NaN
3	2021-04-02	April	Woodland	Helmond	Warandabos	_B4A2402	Wood Anemone	Single flower	NaN
4	2021-04-02	April	Woodland	Helmond	Warandabos	_B4A2403	Wood Anemone	Single flower	NaN
5	2021-04-02	April	Woodland	Helmond	Warandabos	_B4A2404	Wood Anemone	Single flower	NaN
6	2021-04-02	April	Woodland	Helmond	Warandabos	_B4A2405	Wood Anemone	Single flower	NaN
7	2021-04-02	April	Urban	Helmond	industrieterrein	_B4A2406	Red Dead-nettle	Spike	NaN
8	2021-04-02	April	Urban	Helmond	industrieterrein	_B4A2407	Red Dead-nettle	Spike	NaN
9	2021-04-02	April	Urban	Helmond	industrieterrein	_B4A2408	Red Dead-nettle	Spike	NaN

Figure 7. Data Preparation spreadsheet

- **Manual Labeling**

At this point, we started to think about how to prepare the images better for the future model. We read the report of the previous group and approached the situation in a similar way. For the model to work, we needed labeled data and we opted for the most popular way - the program LabelImg. In it the photos (a subset of them) were uploaded and we manually put the coordinates (the labels) of each image. Even though the program is well made, it is still very time consuming and for a week we managed to label only 5 species. As there were images with many flowers in each photo (i.e. cuckooflower or cow parsley), it might not be the optimal solution, thus we decided to take another approach.

- **Automated cropping**

Since our technical tutor (Nico Kuijpers) agreed it indeed requires a lot of time to label the images, he suggested we look up the color detection and cropping code from last year and try to apply it for our set of images. We followed his advice and we quickly noticed that this approach is very fast and might be especially suitable for our case.

The downloading of the small cropped images was done in Matlab so we had to convert the code to Python and make a function. In the beginning we used RGB for the color detection but since it was quite limited from 0 to 255 for the three channels, we switched to HSV. However, this approach was a bit tedious as well because for every flower type we had to use a color picker website to look up the values. Sometime around week 13, we improved the code by adding a trackbar to slide through these color values. This piece of code works by opening 3 instances of the image - the original one, the result one and the mask. With this code it is very easy to be very specific about the color or even choose multiple colors (i.e., yellow and blue). When one is done with filtering the wanted color, the “esc” button is pressed and the 3 opened windows are closed. Then the upper and lower HSV values are printed and can be used for the cropping. Finally, the images are

downloaded in the specified folder with each of them having an additional id number attached to the name. This code does not require a lot of computational power and by using 6 values for color detection it makes it very precise. This notebook can be found here.

3.4 Modeling

When the data was ready, the modeling started where a few of different techniques were selected and applied, and their parameters were tuned in search for the optimal solution.

The duration for this phase was 5 weeks in which the following tasks were completed:

3.4.1 Select modeling techniques

With the data not being labeled, modeling became quite an obstacle in the beginning. We researched different methods for labeling but there is still no automatic way to do so. With the 5 types that we have already manually labeled the Faster R-CNN was built. For the custom model and the one using pre-trained layers, our own cropped images were used.

3.4.2 Build models

- **R-CNN & Faster R-CNN Models**

R-CNN gets around the challenge of selecting a large number of areas, Ross Girshick et al. presented an approach in which we utilize selective search to extract only 2000 regions from an image, which he dubbed region proposals. As a result, rather than attempting to classify a large number of areas, you can now focus on simply 2000.

Faster R-CNN comes from the same author of the previous study (R-CNN) who managed to fix some of R-shortcomings CNN's to create Fast R-CNN, a quicker object detection technique. The method is comparable to the R-CNN algorithm.

Instead of feeding the CNN the region proposals, we give the CNN the input image to produce a convolutional feature map. We select the region of proposals

from the convolutional feature map, warp them into squares, then restructure them into a fixed size using a RoI pooling layer so that they may be fed into a fully connected layer. We utilize a softmax layer to estimate the suggested region's class and offset values based on the RoI feature vector.

For **this project** we have decided to implement the techniques for Faster R-CNN described by Ross Girshick in his research paper. We made that possible by

following the tutorial for setting up the environment and training a Faster R-CNN model demonstrated by Tensorflow-Object-Detection-API-tutorial. Summarized from the tutorial, in order to successfully build and train such model, you would need to have the following requirements :

- Anaconda Python
- Tensorflow pip
- CUDA
- CUDNN
- Tensorflow Model Garden
- Protobuf
- COCO API
- Object Detection API

How to install all the dependencies can be found here and our own code is stored **here**.

Choosing the pre-trained model was an important part for successfully implementing Faster R-CNN. For that we made use of the Faster R-CNN Inception ResNet V2 1024x1024 which can be found in the Tensorflow Object Detection Zoo library.

The **pre-processing** part requires labeling the data with a specific tool used for Faster R-CNN , this tool is called LabelImg and it is also explained in the Tensorflow Object Detection API tutorial along with how to create the training and test records as well as the label_map which are all required as inputs for the model. Additionally to the pre-processing part, we found it useful to resize the images before using the LabelImg tool to **800 - 600**. Otherwise we found issues when training because when trained with the full image resolution we did not have enough computing power (GPU storage).

For training data we used images from only 5 different species :

- Dandelion
- Bistort
- Marsh-Orchid
- Daisy
- Knapweed

Configuration (pipeline.config) is the part where we combine all the requirements and set up the training and model hyperparameters. This is done inside a file that you get after downloading a model, the name of the file is pipeline.config and below we have listed some of the changes we had to perform in order to reach optimal results in terms of accuracy and loss matrices.

Model Hyper-parameters

- Change the image resizer from keep_aspect_ratio_resizer to fixed_shape_resizer with width and height (800 , 600) same as the resized image.
- Change height_stride and width_stride from **16 to 12** (pixels) in that way you can detect smaller objects.
- Change maxpool_kernel_size and maxpool_stride from **1 to 2**
- In the second_stage_post_processing section change the score_threshold and iou_threshold from **(0.0 , 0.6 to 0.1 , 0.1)**
- Change second_stage_localization_loss_weight and second_stage_classification_loss_weight from **(2.0 , 1.0 to 1.5 , 1.0)**
- Add your test.record, train.record, label_map.pbtxt file and fine_tune checkpoint to the required fields

Training Hyper-parameters

```
train_config: {
 batch_size: 1
 num_steps: 15000
 optimizer {
  momentum_optimizer: {
   learning_rate: {
    cosine_decay_learning_rate {
      learning_rate_base: .04
      total_steps: 15000
      warmup_learning_rate: .013333
      warmup_steps: 5000
    }
   }
   momentum_optimizer_value: 0.5
  }
  use_moving_average: false}
```

Note that we managed to get these values by applying the method of test and trial, in other words other possible values for training and model hyper-parameters are also applicable and might provide even better results.

- **Custom Model**

Since we managed to create automated labeling we decided to have our own custom model for this project. In this way, we would be able to compare the results between the models. The model can be found here.

For this custom model we used **CNN**. In order to make use of the labeled folders and images two arrays were created - **x** for feature dataset with the images and **y** for target dataset with labels. Furthermore, the size of the images was decreased to **128x128** because most of the images had different dimensions. The labels were converted to numerical values using **"LabelEncoder()"**. Considering that we want the model to predict the flower species based on the labels, the "y" array holding the labels was converted to one hot encoding. Therefore, the algorithm will have better performance in predicting the labels.
Additionally, the pixels in the images were normalized to be values between 0 and 1. All these steps are essential to have a properly working CNN.

For building the model we used **"Sequential()"** from Keras. It is important to notice that at the beginning we used examples of layers which were common for recognizing flowers. The model does not consist of many layers - 4 convolutional layers, 2 max pooling layers, 1 flatten layer and 2 dense where the last dense layer uses "softmax" activation. However, in the process of adding more species to be trained we customized the parameters, added one more dense layer and a dropout of 50%. For compiling, the **"Adam"** optimizer was applied with a learning rate of 0.0001.

```
model = Sequential()
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu', input_shape = (size,size,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
#model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(30, activation = "softmax"))
```

Figure 8. Custom model specifications

Nevertheless, later we added a checkpoint to monitor the validation loss and save the best results only as well as **"ReduceLROnPlateau()"** in order to reduce the learning rate when the validation loss has stopped improving.

<u>The key things to be taken into consideration are the following:</u>

- The label conversion to tensors is set with a depth of 30 for the 30 species.
- The loss of the model is set to "categorical_crossentropy". For only 2 species, set it to binary_crossentropy.

- The last layer of the model has 30 output neurons for the 30 labels.

It is important to note that the first version of the model was with 2 species (daisy and marsh-marigold). The accuracy was 99% and therefore we added more types -

3, 6, 10, 15, 20, 26 and finally 30. Naturally, the accuracy dropped with the many species, but we are still happy with the results obtained. The types which we trained our model on are: *Bird's-foot Trefoil, Brown Knapweed, Buttercup(Creeping & Meadow), Chamomile, Common Dandelion, Common Poppy, Cornflower, Cow Parsley, Cuckooflower, Field Mouse-ear, Flatweed, Hare's-foot clover, Heath Spotted Orchid, Hoary Alyssum, Lesser Spearwort, Marsh Lousewort, Marsh marigold, Meadow Thistle, Ox-eye Daisy, Perforate St John's-wort, Purple Loosestrife, Ragwort, Red Clover, Redstem Filaree, Southern Marsh Orchid, Tansy, Wild Carrot, White Clover, Yarrow, Yellow Loosestrife.*

Another thing to be considered is that for all these 30 species we cropped about the same number of images (220-240). In the earlier experiments, we had some discrepancies - few of the flower types had 300 cropped photos and few others had 60. This was not good for the model as it led to the mis-classification of some species.

This gradual incrementation can be seen through our regular commits to GitHub. Moreover, we presented our progress to the client every other week.

- **Pre-trained Model (MobileNet V2)**

Due to the validation accuracy of the custom model not being as optimal as we desired it to be, in the last few weeks we decided to experiment with a pre-trained model. We had a similar individual task for a CNN with a pre-trained model in the beginning of the semester, so we decided to give it a try.

Making use of pre-trained models saves a lot of time and computational budget for our classification problem. The MobileNet_V2 model was used as the architecture delivers high accuracy results while keeping the parameters and mathematical operations as low as possible.

```
# having the model ready
classifier = tf.keras.Sequential ([
   hub.KerasLayer("https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/3", input_shape = size +(3,))
])
```

Figure 9. Pre-trained model loading

The fact that our custom model was ready, it was quite straightforward to train this model as well. The loading of the image dataset is the same, its preparation also. When the MobileNet_V2 model is loaded and the data is ready, it is crucial to introduce the feature vector. Its aim is to give all the layers of the pre-trained model except the last one which is determined by us. The “trainable” property has to be set to false which means to freeze and not train (all the layers will have their fixed

weights). The model is compiled in the standard way and the code can be seen here.

```
    # the feature vector gives all the layers except the last one
feature_extractor_model = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/3"
```

```
# trainable false means freeze, do not train (all the layers will have their fixed weights)
pretrained_model_without_top_layer = hub.KerasLayer(
    feature_extractor_model, input_shape=(224, 224, 3), trainable=False)
```

```
num_of_flowers = 30

# create the last layer
model = tf.keras.Sequential([
  pretrained_model_without_top_layer,
  tf.keras.layers.Dense(num_of_flowers)
])

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer_1 (KerasLayer)   (None, 1280)              2257984
_________________________________________________________________
dense (Dense)                (None, 30)                38430
=================================================================
Total params: 2,296,414
Trainable params: 38,430
Non-trainable params: 2,257,984
_________________________________________________________________
```

Figure 10. Pre-trained model specifications

The key things to be taken into consideration are the following:

- The images were resized to 224 x 244, instead of 128 x 128, but the latter option could also work. We opted for the first one because the model was well trained with only 10-15 epochs so the higher quality and size, the better.
- For this model we do not need the transformation to tensors, thus this step is skipped.
- The last layer of the model has 30 output neurons for the 30 labels (num_of_flowers in our case).
- The one_hot tensor was only needed for the test labels and the evaluation.

3.5 Evaluation

In this phase, the final evaluation and reviewing of the models is done. Here, the importance lays on the business value again and if everything has been sufficiently considered.

The duration for this phase was 2 weeks in which the following tasks were completed:

3.5.1 Evaluate results

- **Faster R-CNN**

After training for 15,000 steps on 5 different flower types using the Faster R-CNN Inception ResNet V2 1024x1024 model we get the following evaluation for the loss matrix using Tensorboard.
In general we see a steady decrease in loss over time, with an average of 0.05 - 0.1 for classification loss and 0.12 - 0.2 for localization loss.

Figure 11. Faster R-CNN evaluation

- **Custom model**

The model was trained firstly on 64 epochs which were reduced to 50 when the model was having too much validation loss considering the increasing amount of data.

```
Epoch 00043: val_loss did not improve from 0.81528
Epoch 44/50
117/117 [==============================] - 174s 1s/step - loss: 0.5047 - accuracy: 0.8105 - val_loss: 0.9067 - val_accuracy: 0.
7756

Epoch 00044: val_loss did not improve from 0.81528
Epoch 45/50
117/117 [==============================] - 177s 2s/step - loss: 0.4536 - accuracy: 0.8316 - val_loss: 0.9142 - val_accuracy: 0.
7724

Epoch 00045: val_loss did not improve from 0.81528
Epoch 46/50
117/117 [==============================] - 176s 2s/step - loss: 0.4404 - accuracy: 0.8351 - val_loss: 0.9465 - val_accuracy: 0.
7830

Epoch 00046: val_loss did not improve from 0.81528
Epoch 47/50
117/117 [==============================] - 174s 1s/step - loss: 0.4244 - accuracy: 0.8447 - val_loss: 1.0345 - val_accuracy: 0.
7662

Epoch 00047: val_loss did not improve from 0.81528
Epoch 48/50
117/117 [==============================] - 181s 2s/step - loss: 0.4247 - accuracy: 0.8383 - val_loss: 1.0474 - val_accuracy: 0.
7481

Epoch 00048: val_loss did not improve from 0.81528
Epoch 49/50
117/117 [==============================] - 194s 2s/step - loss: 0.4286 - accuracy: 0.8437 - val_loss: 0.8481 - val_accuracy: 0.
7905

Epoch 00049: val_loss did not improve from 0.81528
Epoch 50/50
117/117 [==============================] - 198s 2s/step - loss: 0.4317 - accuracy: 0.8386 - val_loss: 1.1864 - val_accuracy: 0.
7693

Epoch 00050: val_loss did not improve from 0.81528
```

Figure 12. Custom model epochs

Taking into account that there are 30 flower species the model gives reliable results despite that the validation loss is increasing exactly in the last few epochs. The accuracy has reached **76%** which is understandable from one side because there are few species that look similar to each other even for the human eye.

From the confusion matrix below we can see that the most mistaken species are Red Clover with Brown Knapweed, Tansy with Yellow Loosestrife and Hares-Foot Clover with Hoary Alyssum. Even though there are approximately 4950 images, some of which are not enough for the model to learn and distinguish the species, we believe that this model is appropriate to be used for the future.

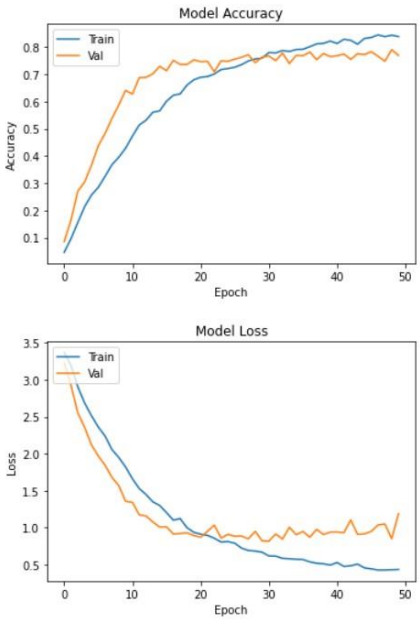

Figure 13. Model Accuracy & Loss

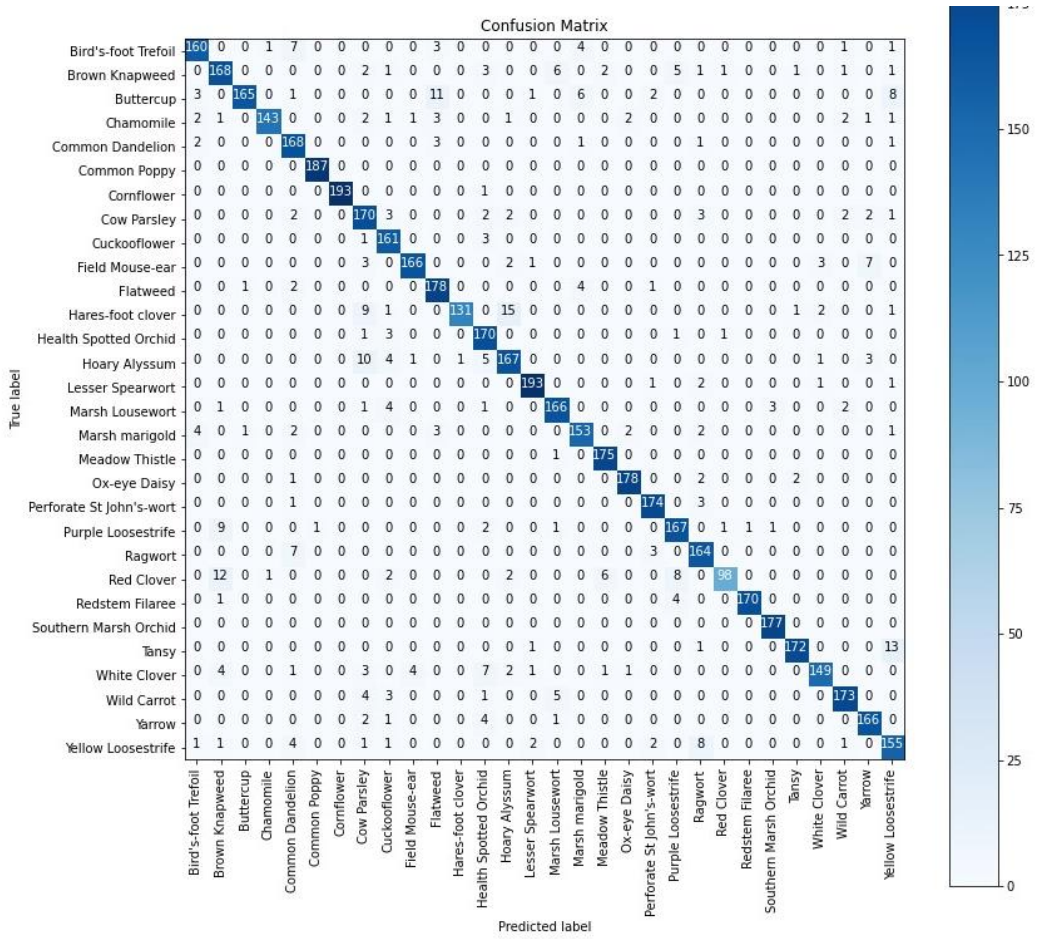

Figure 14. Custom Model Confusion Matrix

- **Pre-trained Model (MobileNet_V2)**

The results from the pre-trained model look very promising. For 10 epoch, the model was able to achieve validation accuracy of 98% which we were lacking with our custom model.

```
Epoch 00005: val_loss improved from 0.13933 to 0.12519, saving model to model.h5
Epoch 6/10
134/134 [==============================] - 57s 429ms/step - loss: 0.0551 - accuracy: 0.9970 - val_loss: 0.1098 - val_accuracy: 0.9794

Epoch 00006: val_loss improved from 0.12519 to 0.10976, saving model to model.h5
Epoch 7/10
134/134 [==============================] - 56s 421ms/step - loss: 0.0428 - accuracy: 0.9986 - val_loss: 0.1011 - val_accuracy: 0.9804

Epoch 00007: val_loss improved from 0.10976 to 0.10108, saving model to model.h5
Epoch 8/10
134/134 [==============================] - 56s 421ms/step - loss: 0.0353 - accuracy: 0.9984 - val_loss: 0.0971 - val_accuracy: 0.9785

Epoch 00008: val_loss improved from 0.10108 to 0.09709, saving model to model.h5
Epoch 9/10
134/134 [==============================] - 57s 427ms/step - loss: 0.0290 - accuracy: 1.0000 - val_loss: 0.0937 - val_accuracy: 0.9804

Epoch 00009: val_loss improved from 0.09709 to 0.09371, saving model to model.h5
Epoch 10/10
134/134 [==============================] - 56s 421ms/step - loss: 0.0241 - accuracy: 0.9998 - val_loss: 0.0885 - val_accuracy: 0.9813

Epoch 00010: val_loss improved from 0.09371 to 0.08850, saving model to model.h5
```

FIgure 15. Pre-trained model epochs

The model accuracy increases rapidly after the first few epochs and the loss decreases respectively. This is a common thing that happens with the pre-trained models.

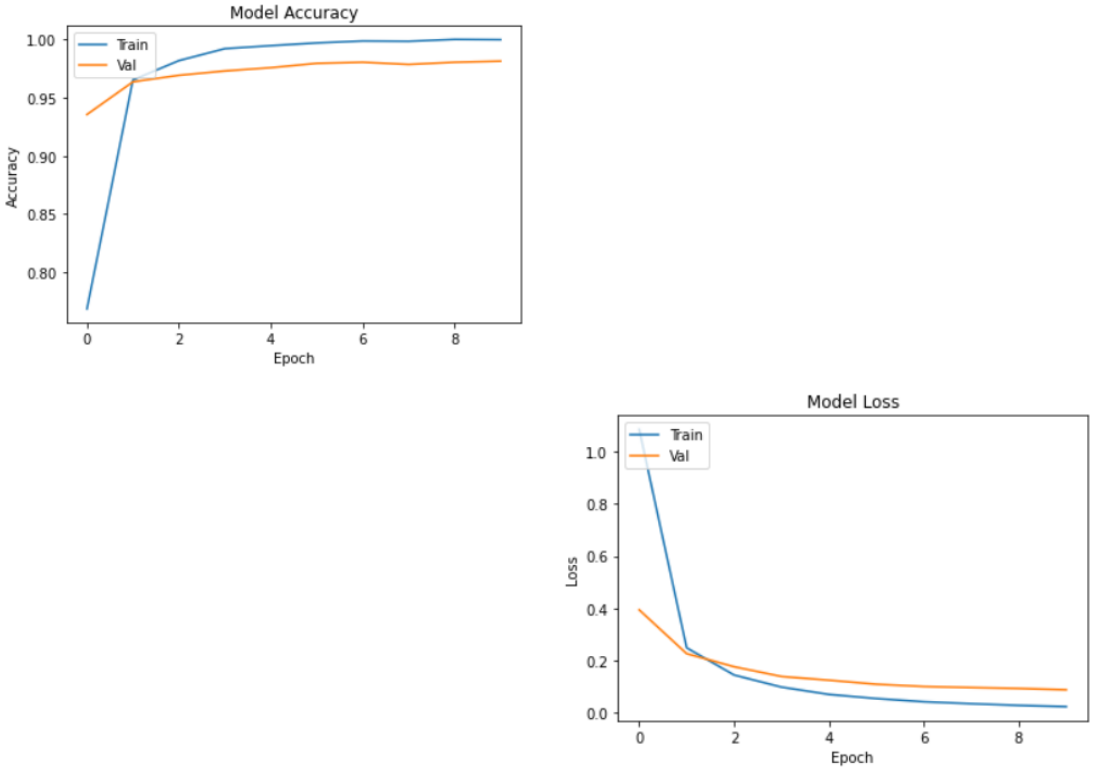

Figure 16. Pre-trained model Accuracy & Loss

Below, the confusion matrix can be observed. In it, we can clearly see that there are extremely few mistakes. The maximum mis-classifications are 4, which is very low given the number of labels and the 10 epochs.

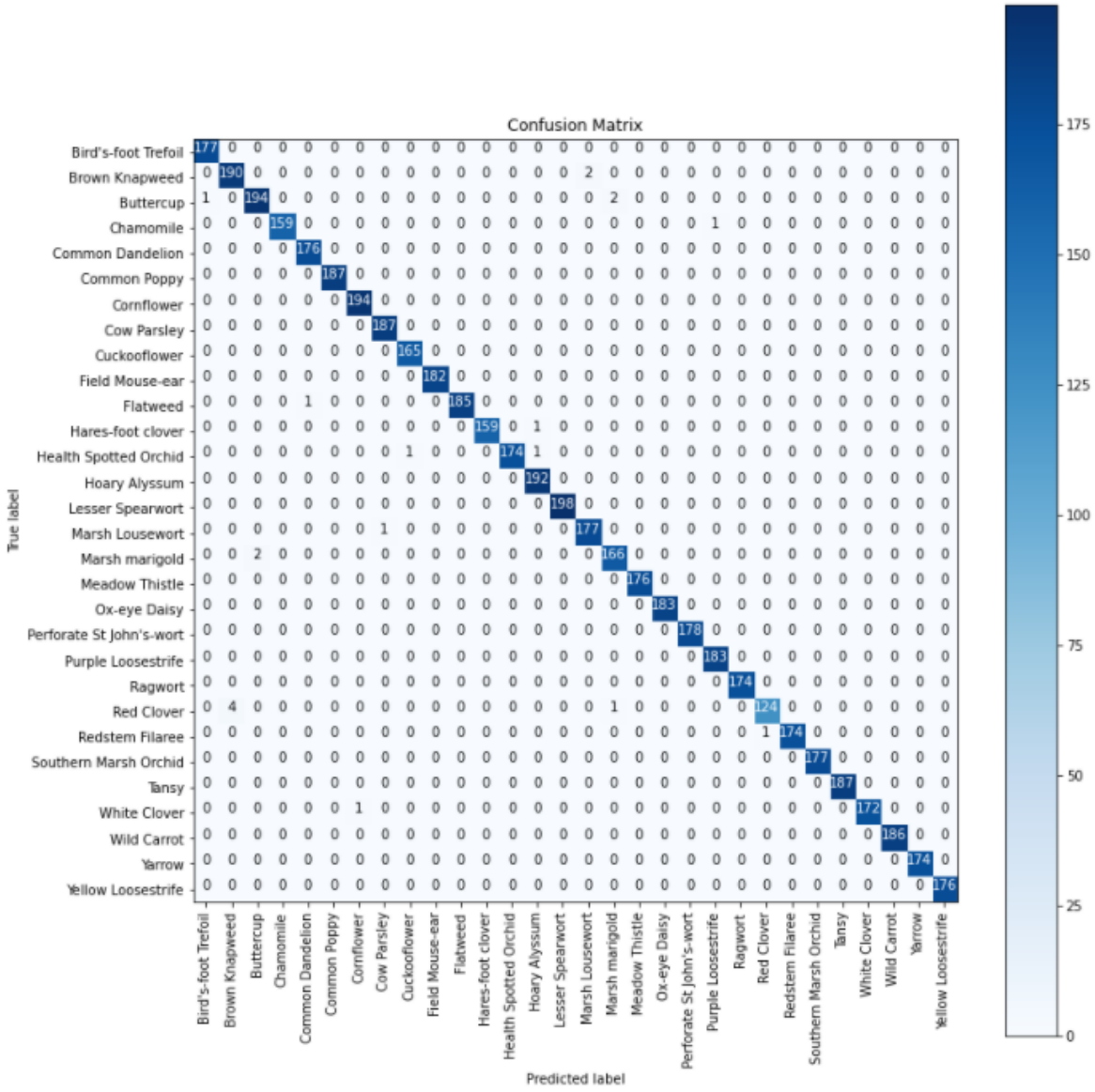

Figure 17. Pre-trained model confusion matrix

3.5.2 Review process

The three approaches gave impressive results and thanks to the iterative methodology that we have chosen, we managed to improve and expand the models effortlessly. Overall, our group worked very hard to have not only two, but three different more than sufficient models and we are extremely satisfied with the results achieved.

3.6 Deployment

Usually, the final step is to launch the model. However, we are more focused on delivering the model in such a way that it can be understood and used by future groups or to be integrated in other similar Fontys research projects.

The duration for this phase was 2 weeks in which the following tasks were completed:

- **Produce Final Report**

During the last 2 weeks, we were focused on writing this report and the accompanying two-page mini paper. They were written by all team members and their purpose is to nicely wrap-up the whole project with its processes, the results, explanations and recommendations.

- **Final Presentation**

Along with the documents, we prepared a presentation for our client, mentors and fellow students which again summarizes the Flower Power III project.

- **Upload Project to DEX**

The transfer of the code is of vital importance for the successful finish of this project. As we stored the project in GitHub, this is the main place where it can be found. Both our technical tutor and our client have access to the repository. In addition to it, we have uploaded our project to DEX - a platform by Fontys where projects can be uploaded. The link to the DEX project.

4. Results

- **Faster R-CNN**

After the model is trained a set of checkpoints is generated with the final checkpoint being the “most” trained version. For loading the model we have created a notebook in Jupyter where you can test your localization and classification functions. The input is an image and the output is an image with bounding boxes showing the location and type of flower that has been detected by the model. Getting the number and type of detections is also available in the notebook. Here are examples of some of the results we got from the training data (images):

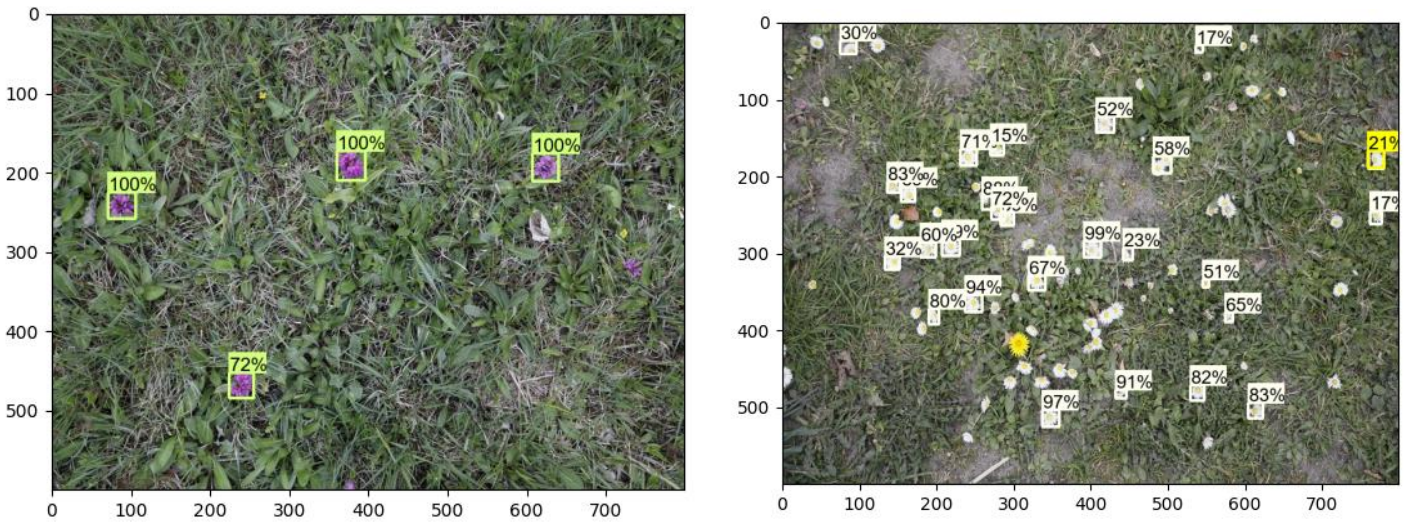

Figure 18. Faster R-CNN results

- **Custom Model**

The CNN model reached validation accuracy of 76% for 30 flower species. Despite that we evaluated the model by plotting the validation accuracy and loss as well as a confusion matrix, we plot an evaluation on the dataset. This is done by displaying an image from the dataset with the true label above and the predicted label by the model. Moreover, when the model predicts correctly the labels above and below the image are colored in green and if it is incorrect in red.

In this way we can see clearly which flower species are mistaken. Here 36 images are shown from which only 11 are not predicted correctly. That is approximately 30% wrong predictions which is not very much taking into consideration that there are few species which are very similar to be distinguished by a computer.

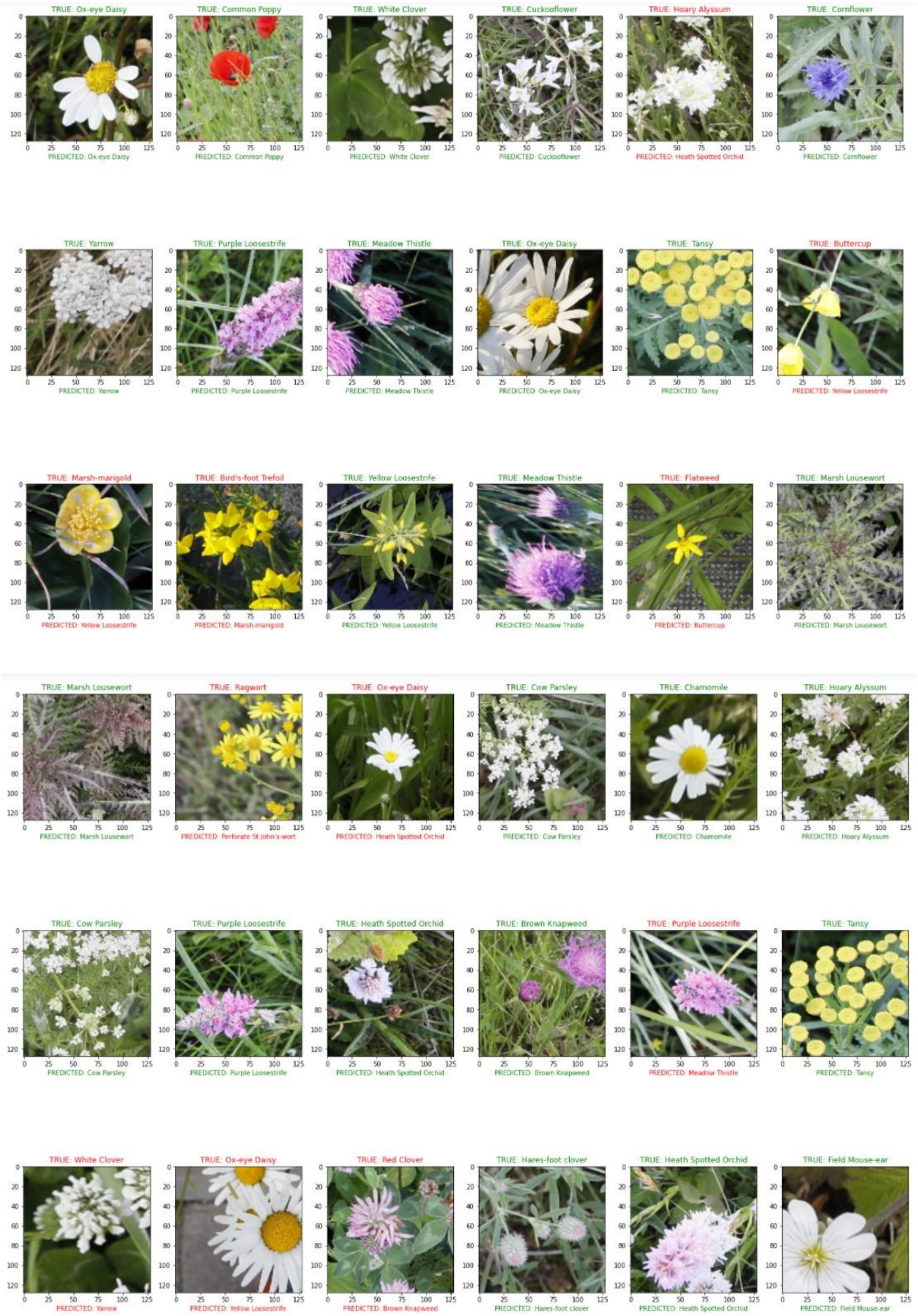

Figure 19. Custom model results

- **Pre-trained model (MobileNet_V2)**

As mentioned in the previous section, the pre-trained model ended up with an accuracy score of 98% for the 30 species. The same approach as the custom model was used to visually show how the model performed on the test set. The 36 images served as a random subset of the whole test set for the evaluation. With such high accuracy it was no surprise that there were no photos mistaken.

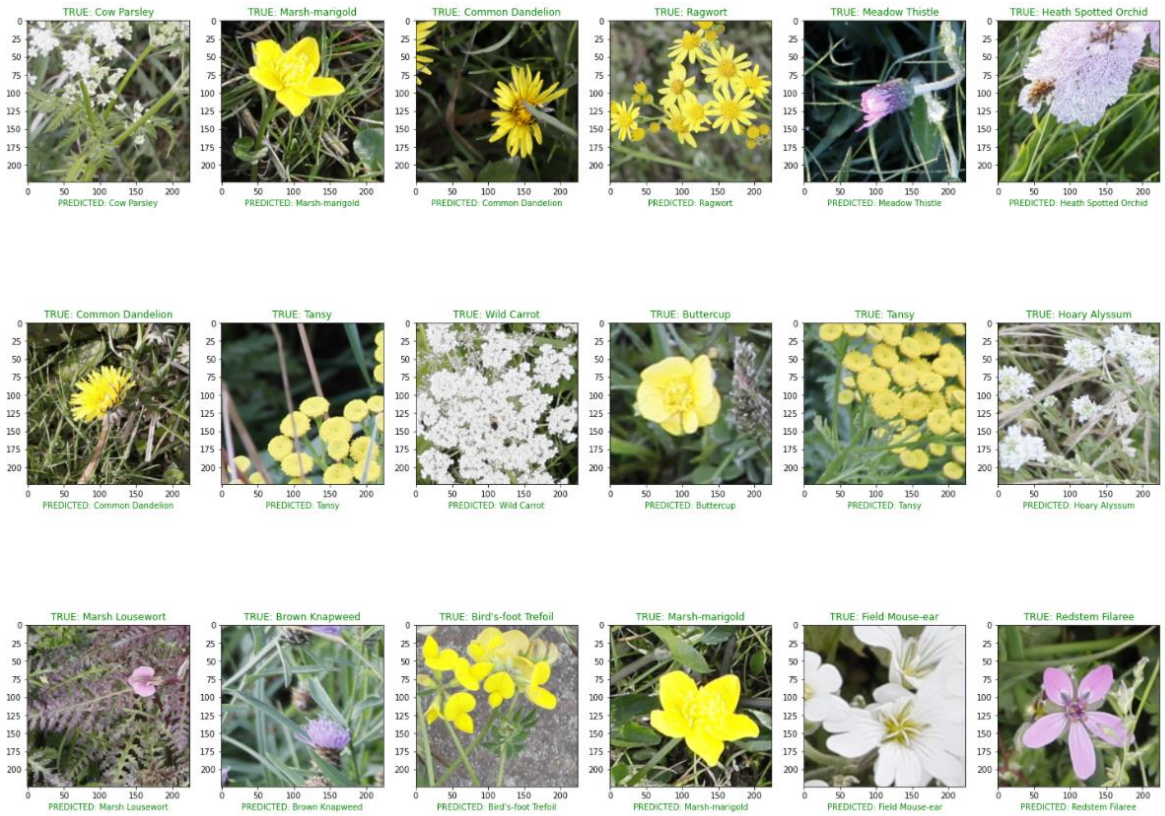

Figure 20. Pre-trained model result

5. Ethical Consideration

Below you can find the Quick Scan made with the help of the Technology Impact Cycle Tool (TICT) framework which concerns the ethical aspect of the project.
The ethical aspects which need to be considered regarding this project are not many, as the data which would be used does not contain any sensitive personal data. The first and of utmost important ethical aspect which needs to be taken into consideration is the potential usage of drones, as the initial idea of the client is to use those devices to collect images of flower species. Therefore, he needs to obtain the corresponding permission from the local authorities. Furthermore, the client needs to make sure that if in the area where he wants to deploy the drone there are people residing, he needs to obtain their permission beforehand. Additionally, the client should also consider making public information the times during which the drone will be active as well as in which areas. Furthermore, many researches have shown that the deployment of drones can often lead to a negative impact on society.

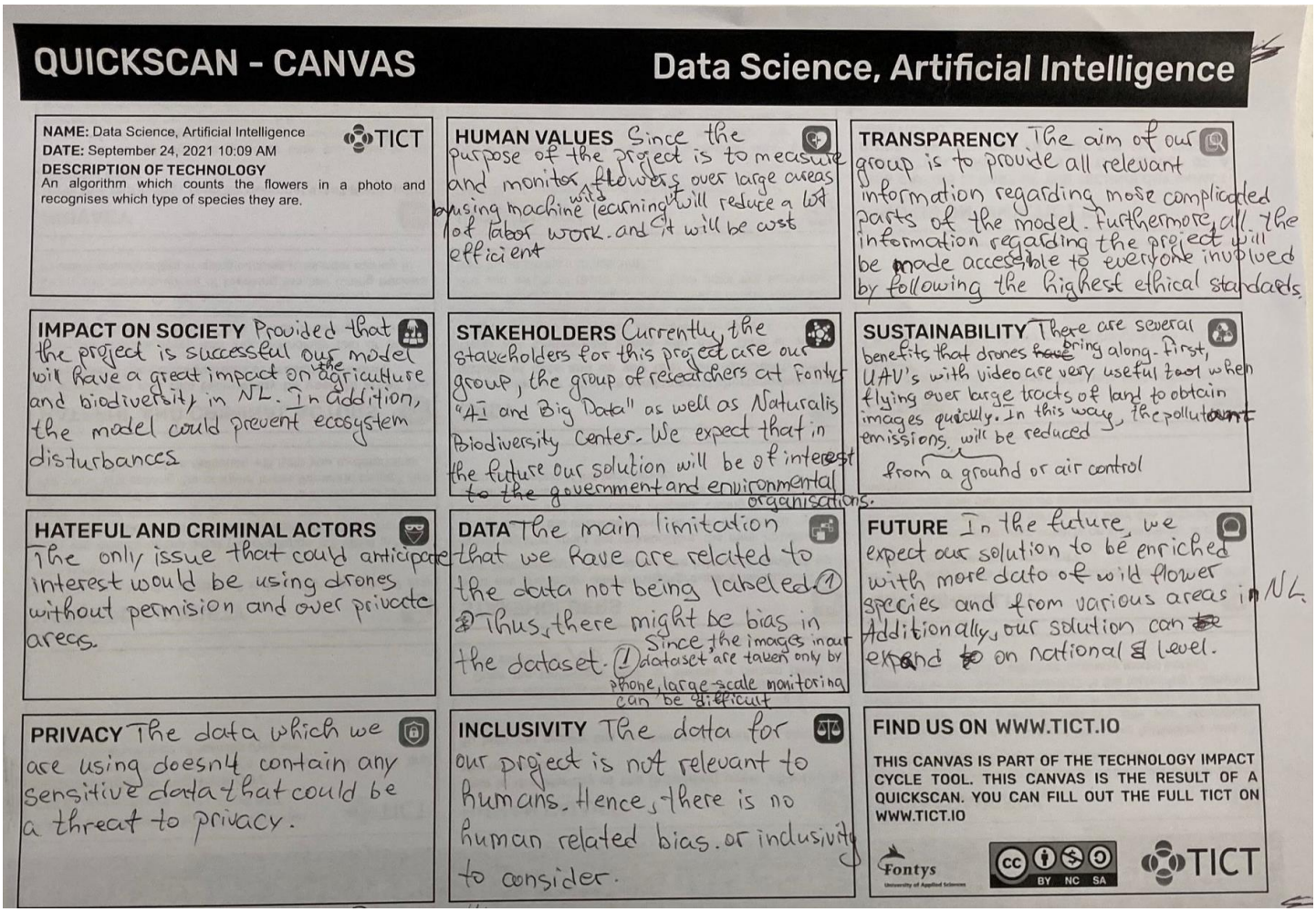

Figure 21. Flower Power Quick Scan

6. Conclusion

For this project we ended up with two approaches of formatting the data - manual labeling and automated cropping. The manual labeling is necessary for the Faster R-CNN model using the tool LabelImg. Here the disadvantage is that the labeling takes a longer time for which we labeled only 5 species of wild flowers. The automated cropping was taken from last year's Flower Power project. However, the downloading of the small cropped images was done in Matlab so we had to convert the code to Python and make a function using RGB color detection. Soon after that we switched to HSV color detection which was more useful for the custom and pre-trained model.

Choosing the pre-trained model was an important part for successfully implementing Faster R-CNN. For that we made use of the Faster R-CNN Inception ResNet V2 1024x1024. Additionally to the pre-processing part, we resize the images before using the LabelImg tool to 800 - 600. Otherwise we found issues when training because when trained with the full image resolution we did not have enough computing power (GPU storage). As a result from the Faster R-CNN model is an image with bounding boxes showing the location and type of flower that has been detected by the model. Getting the number and type of detections is also available in the notebook.

The custom model that uses CNN was firstly trained on 2 species which resulted in validation accuracy of 99%. With time the number of species was increasing and the accuracy was decreasing. Overall, the model is trained on 30 species with an accuracy of 76%. Even though the model is able to distinguish flowers which are pretty similar between each other and during the evaluation 70% of the predictions were correct.

Nevertheless, since the validation accuracy of the custom model was not as optimal as we desired it to be, we experimented with a pre-trained model. The MobileNet V2 model was used as the architecture delivers high accuracy results while keeping the parameters and mathematical operations as low as possible. The pre-trained model accomplished an accuracy score of 98% for the 30 species. With such high accuracy it was no surprise that there were no photos mistaken during the evaluation on the test set in comparison with the custom model.

To conclude, each of the three models has its own benefits. Depending on the time and computational consumption we as a group provided solutions which are flexible and will be useful for smaller or bigger datasets. Each model can be applied according to the available resources which are needed for the performance of the models.

7. Recommendations

Faster R-CNN

The implementation of Faster R-CNN is a convenient choice for the goal of the project since it provides all the requirements such as counting and classifying flowers all together. But the current version is far from perfect since we noticed a steady amount of flowers that are **not detected or misclassified**. The approach is correct but there are several things that can be changed to provide better results. Based on the project report An Optimized Deep Learning Model For Flower Classification Using NAS-FPN And Faster RCNN which covers the same type of project as Flower Power III but with much higher accuracy when it comes to results. We discovered several aspects that must be considered for future development:

- **More evenly distributed data.** One of the major reasons why Faster R-CNN was harder to implement for this project is the fact that the data had to be labeled manually. Except that the number of different flower types was not close to each other. making it impossible to implement for all different flower types.
- **More computing power.** The model was trained on a Nvidia GTX 1060 with dedicated storage of 6 GB. Even though it was enough for training it required downsizing the original image resolution to (800-600) which will affect the accuracy of the model. It also makes it harder because it requires on average 4-5 hours of training for 15,000 steps where usually a model like this would be trained for 100,000 steps.

Custom model

In order to improve the CNN model it will be better to have more images of the flowers. Right now there are species which have very small numbers of images in comparison where others have too much. In this way when the images are cropped the parameters have to be adjusted depending on the flower. Therefore, having enough images of each flower the model will be able to learn faster and more efficiently, to better distinguish species with similar shape and/or color and achieve higher accuracy.

Pre-trained model (MobileNet_V2)

Since this model performed extremely well, we would recommend for any future groups to experiment with it even further. Having more images and flower species would be an interesting continuation to the work already done. In this way it would be determined whether or not the pre-trained model is actually that good. Another thing that can be considered is to try a different pre-trained model and see if the results are any better. The last recommendation would be to try any of the

mentioned models with pictures of drones as this is the original primary focus of Flower Power.

8. Reflection

During the ADS Minor, our group had the pleasure to work with our technical tutor Nico Kuijpers and our client Gerard Schouten, both of whom provided valuable and extensive feedback.

We approached the Flower Power project with a lot of enthusiasm, interest and willingness to learn new techniques. As all of us are passionate about AI it came as no surprise that we managed to execute this project successfully. The communication within the team was effortless, the distribution of the tasks was equal and the delivered products more than satisfactory. Each group member gave feedback to their fellow students and in this way every one of us was able to improve themselves.

Thanks to the fact that almost all of the semester was on site, we had frequent meetings and on many occasions we worked together - exchanging ideas, suggesting improvements, peer coding and reviewing, working towards the ideal solution and achieving our project goal. Because of our planning being detailed and always followed dutifully, we did not face any complications and all deliveries were submitted before the deadline. Additionally, our eagerness to improve after each iteration contributed to us managing to provide multiple solutions.

Looking back on it, we are glad that we did not spend a considerable amount of time focusing on one task, for instance, the manual labeling of the images. Instead, we were flexible and tried another approach - the automated cropping which turned out to be highly beneficial for 2 of our CNN models. Nevertheless, all three methods were very well received by our mentors and the client. We were able to learn a great deal out of them - how to set up the environment, clean and prepare the data and run numerous modeling experiments in the name of achieving an optimal solution.

We are more than happy that we had chosen Flower Power III as our semester project as it was favorable and valuable not only for our technical skills, but it was also an excellent example of how when communication and planning are effective, we can only reap the benefits of our work.

References

1. Following the data science methodology. (n.d.). IBM Developer. https://developer.ibm.com/blogs/following-the-data-science-methodology/
2. Step-by-step data mining guide. (2000). Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. https://the-modeling-agency.com/crisp-dm.pdf
3. Gandhi, R. (2018, July 9). *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms*. Towards Data Science; Towards Data Science. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
4. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf
5. *Training Custom Object Detector — TensorFlow Object Detection API tutorial documentation*. (2017). Readthedocs.io. https://tensorflow- object-detection-api-tutorial.readthedocs.io/en/latest/training.html
6. *Installation — TensorFlow Object Detection API tutorial documentation*. (2019). Readthedocs.io. https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html
7. Patel, I., & Patel, S. (n.d.). *An Optimized Deep Learning Model For Flower Classification Using NAS-FPN And Faster R- CNN*. Retrieved January 12, 2022, from http://www.ijstr.org/final-print/mar2020/An-Optimized-Deep-Learning-Model-For-Flower-Classification-Using-Nas-fpn-And-Faster-R-cnn.pdf
8. *Why Google's MobileNetV2 Is A Revolutionary Next Gen On-Device Computer Vision Network*. (2018, April 18). Analytics India Magazine. https://analyticsindiamag.com/why-googles-mobilenetv2-is-a-revolutionary-next-gen-on-device-computer-vision-network/
9. Kharwal, A. (n.d.). *Flower Recognition with Python*. Data Science | Machine Learning | Python | C++ | Coding | Programming | JavaScript. https://thecleverprogrammer.com/2020/11/24/flower-recognition-with-python/