École Polytechnique Fédérale de Lausanne

GrowBotHub : Computer Vision Flower Object Detection

Student : Emna Maghrebi
Supervisor : Joachim Ludovic Hugonot

Bachelor Semester Project

Abstract

With the recent advancement of deep learning, the performance of object detection techniques has greatly increased in both speed and accuracy. This has made it possible to run highly accurate object detection with real time speed on modern desktop computer systems. Recently, there has been a growing interest in developing smaller and faster deep neural network architectures suited for embedded devices. This project explores the suitability of running object detection on the vegetable flowers that GrowBotHub is seeking to plant .

The object detector that has been chosen is faster RCNN due to its widespread use and highly accurate object detection reported for various applications . Using the method of transfer learning, different pre-trained models including ResNet-18, ResNet-50 and ResNet-152 are trained and evaluated on a Synthetic flower image dataset that contains different flower images.

Keywords: computer vision, object detection, vegetable flower

Contents

1. Introduction

Computer vision is a field of computer science which enables machines not only to see but also to process and analyze digital images and videos . One big application area of computer vision is object detection which is the task of recognizing and localizing different objects, normally in images depicting daily-life objects.

In recent years, flower detection has been of considerable interest to the computer vision community .
More specifically , The GrowbotHub project , which was founded by a team of EPFL/UNIL students from the Légumes Perchés Association and took part in the first edition of Igluna, an international student project to build a habitat demonstrator for sustaining life in an extreme environment such as on the Moon , aims to grow vegetables autonomously by robot pollination. In order to do that , flower object detection of the flowers in the smart garden must be achieved.

In this project , the problem of object detection in the context of a virtual environment is investigated, where the objects of interest were represented by approximately 25 different flower types.
The purpose of the next chapter is to present the reasoning behind the problem investigated , why the work in this project was conducted and the reasoning behind the research questions stated.

2. Background / Related Work

The field of computer vision has experienced substantial progress recently, owing largely to advances in deep learning, specifically convolutional neural networks (CNNs) .
The ultimate purpose of object detection is to locate important items, draw rectangular bounding boxes around them, and determine the class of each item discovered.

Researchers have dedicated a substantial amount of work towards this goal over the years and were struggling to find a solution to image classification with very low error rate.
But thanks to Alex Krizhevsky who presented the CNN AlexNet (The First CNN to win ImageNet) at the ImageNet Large Scale Visual Recognition Competition in 2012 , many object detecting methods applying CNN have been presented showing great performance and efficiency. Newer and better versions of this CNN were VGG16 and ResNet .

Applications of object detection arise in many different fields including detecting pedestrians for self-driving cars, monitoring agricultural crops..
Plant identification is an important task for researchers, students, and practitioners in field of the agriculture, forest, biodiversity protection, and so on . In our project we're interested in identifying the flowers in order to pollinate them , which is the main motivation to use Object detection for this task .

A very recent study : *DNN Based Real-time Kiwi Fruit Flower Detection* has been made to build an accurate, fast, and robust autonomous pollination robot system and where they trained and compared different models (with Faster R-CNN and Single Shot Detector (SSD) Net) for kiwi fruit flower detection .

Also , an interesting study : *Object Detection for Flower Recognition* was conducted in EPFL which used Faster R-CNN in order to do object detection on pictures of flower bouquets , pass them through an algorithm , and it would recreate the bouquet digitally .

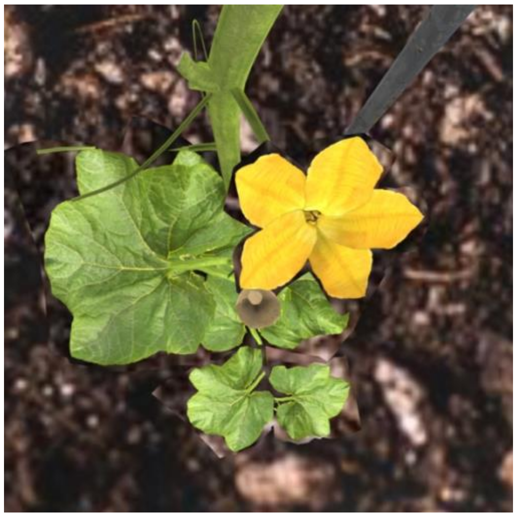
Inspired by this, we aim for exploiting DNNs for vegetable flower detection and present experiments and their analysis on object detector Faster R-CNN and feature extractors ResNet18, ResNet 50 and ResNet 152 with real-world synthetic datasets.

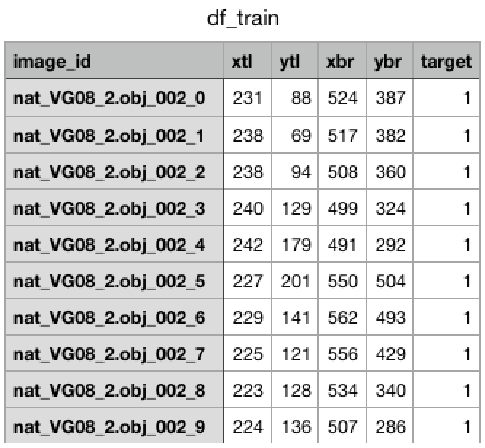
3. Methodology

Dataset generation

For this project we had to use synthetic Data since adequate flowers dataset could not be found in order to make a good training for object detection. In fact , I have searched various websites that contain Datasets like Kaggle , Roboflow.. I also contacted **Zhibin Cheng** and **Fuquan Zhang** for their paper Flower End-to-End Detection Based on YOLOv4 to ask if they could provide me their annotated dataset of flowers , but I didn't receive an answer from them.

The dataset consisted of a set of synthetic images of flowers with corresponding metadata stored as CSV files containing information about the flowers that are inside.





df_train

image_id	xtl	ytl	xbr	ybr	target
nat_VG08_2.obj_002_0	231	88	524	387	1
nat_VG08_2.obj_002_1	238	69	517	382	1
nat_VG08_2.obj_002_2	238	94	508	360	1
nat_VG08_2.obj_002_3	240	129	499	324	1
nat_VG08_2.obj_002_4	242	179	491	292	1
nat_VG08_2.obj_002_5	227	201	550	504	1
nat_VG08_2.obj_002_6	229	141	562	493	1
nat_VG08_2.obj_002_7	225	121	556	429	1
nat_VG08_2.obj_002_8	223	128	534	340	1
nat_VG08_2.obj_002_9	224	136	507	286	1

In the CSV file, the image_id is corresponding to an image name (in the example above, the image is called **nat_VG08_2.obj_002_0** and it's a .png picture). Each flower is represented by a bounding box whose bounds coordinates are respectively **xtl** , **ytl** , **xbr** , **ybr** fields of the CSV , **tl** meaning top left and **br** meaning bottom right . While generating the dataset , we also use information about the center and stem coordinates of the flower , but they don't figure on the CSV file since we don't need it for detection.

The images were generated by **Pyrender** for physically-based rendering and visualization.
Since the images were computer-generated, the bounding boxes were directly computed by a 2D projection to the camera . Thus, it was not necessary to have humans do the tedious work of detouring and labeling flowers. We tried to make the images look as realistic as possible , even though they are rendered .

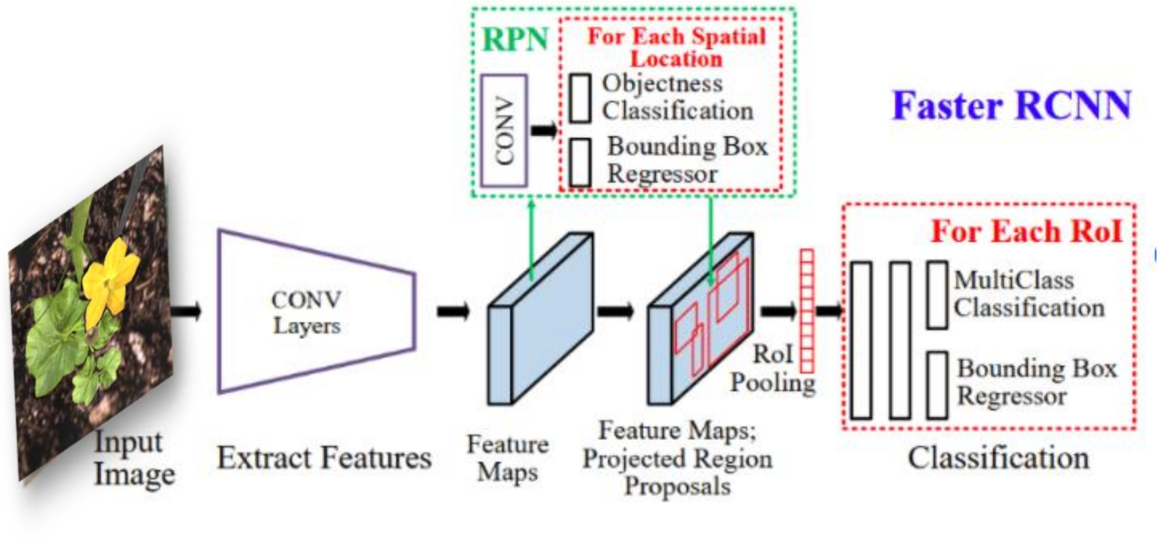
The flowers were also added on random backgrounds so that the neural networks were not used to having a white background which is not the case of a real-life application where we detect the flowers from a garden for example.

Faster R-CNN details

As stated earlier, I chose to implement Faster R-CNN. I thought about implementing YOLO (You only look once) at first , but then I realized it's not the best suit for this problem for different reasons. For example , YOLO has lower precision but smaller computation time as it is designed for video and real-time detection application. The prediction time is not a problem for our case since we're not using it on a video . Also , YOLO struggles on small objects and crowded images.

Thus, Faster R-CNN was a better option , as I will show more in detail how it works .

Faster R-CNN is a meta-architecture that generates box proposals using neural networks, as shown in the figure below . It additionally introduced the Region Proposal Network (**RPN**), which shares convolutional features with the classification network, and two networks are concatenated as one network that can be trained and tested through an end-to-end process.



Faster R-CNN consists of two parts: a region proposal network (RPN) and a region classifier. In the first stage, images are processed by a feature extractor, such as VGG16 and ResNet-18, and features are used to predict class box proposals. In the second stage, these box proposals are used to crop features from the same feature map which are then fed to the remainder of the feature extractor to predict a class and class-specific box refinement for each proposal.

Residual neural network (ResNet)

In this project I have used pre-trained ResNet18 , ResNet50 and ResNet152 networks as feature extractors .
ResNet , short for Residual Network is a specific type of neural network that was introduced in 2015 by *Kaiming He*, *Xiangyu Zhang*, *Shaoqing Ren* and *Jian Sun* in their paper "Deep Residual Learning for Image Recognition". ResNet has won several competitions and its architecture allows for better learning in deeper networks .

It is mainly consisting of convolutional and identity blocks. There are many variants of ResNets, for instance ResNet-50 which is composed of 26 million parameters, ResNet-101 with 44 million parameters and ResNet-152 which is deeper with 152 layers. ResNet-50 and ResNet-101 are used widely in object detection models.

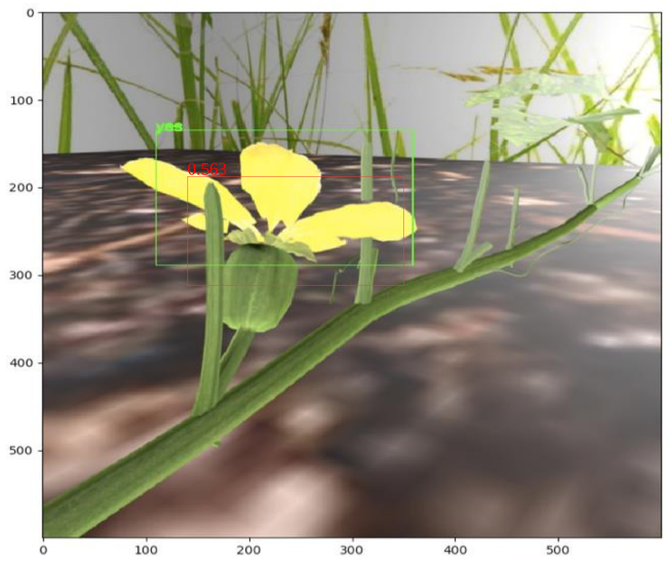
intersection-over-union

Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.
The similarity is expressed in terms of aspect ratio and area. The formula is

$$IoU(A,B) = \frac{area(A \cap B)}{area(A \cup B)}$$

which tells how much two boxes overlap on a scale from 0 to 1.

In the numerator we compute the ***area of overlap*** between the *predicted* bounding box and the *ground-truth* bounding box.
The denominator is the ***area of union*** which is the area encompassed by *both* the predicted bounding box and the ground-truth bounding box.
We can see that if A and B do not overlap, $area(A \cap B)$ is 0, so the IoU will be 0, similarly, if A and B are exactly the same, $A \cap B$ and $A \cup B$ are the same so the IoU will be 1.



4. Implementation

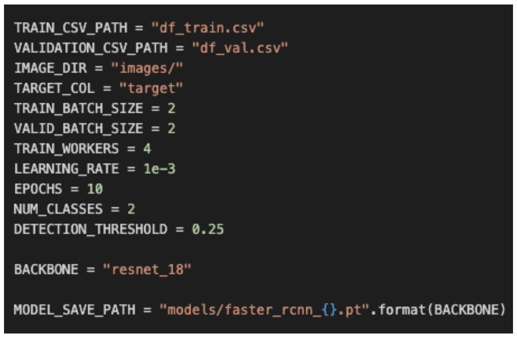
Implementing the network structure was a bit hard, especially because I didn’t have any previous experience with PyTorch before . This took some time because I had not much references other than the papers presenting Faster R-CNN, which leaves some parts ambiguous.

The software in this project consists of scripts written in the Python programming language which is one of the most popular languages used for machine learning tasks. There exist several open source deep learning frameworks that can be used for object detection but for this project, the open source computer vision library PyTorch has been chosen to handle the task.

Scripts

config.py

This script contains the configuration files for training and dataloader , such as the training/validation CSV files , all the image directories such as input and output images , the detection threshold , the backbone name , and all other useful paths.
This configuration file can be edited as per one’s needs .



```
TRAIN_CSV_PATH = "df_train.csv"
VALIDATION_CSV_PATH = "df_val.csv"
IMAGE_DIR = "images/"
TARGET_COL = "target"
TRAIN_BATCH_SIZE = 2
VALID_BATCH_SIZE = 2
TRAIN_WORKERS = 4
LEARNING_RATE = 1e-3
EPOCHS = 10
NUM_CLASSES = 2
DETECTION_THRESHOLD = 0.25

BACKBONE = "resnet_18"

MODEL_SAVE_PATH = "models/faster_rcnn_{}.pt".format(BACKBONE)
```

train.py

This script is responsible for training the dataset. First , it creates the training dataset and validation dataset , along with the training and validation dataloaders . Then , for a specific number of epochs (here it’s configured to 10) , we run the training function and determine the training loss value for each epoch .
Afterwards , the trained model is saved to Disk in the location **MODEL_SAVE_PATH .**

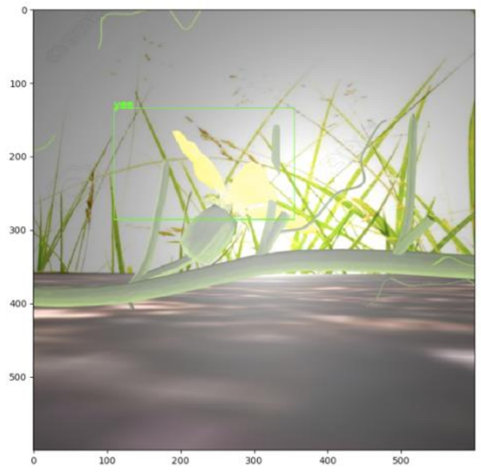
inference.py

This script runs the detector, using any of the specified models. The first thing it does is to check if there is an available GPU else it runs it on the CPU.
Then it loads the model and proceeds with getting the prediction , and then saving the predicted image to the output file.
Before showing the final predictions, the detector goes through a step of filtering to remove poor detections , which consists of removing predictions with a score lower than the confidence threshold value. Setting this value too low will lead to false detections being included. This value is set to 0.25 which means that any detection with a confidence score lower than 25% is not counted as a detection.
When the filtering is done, bounding boxes are drawn around the detected objects .

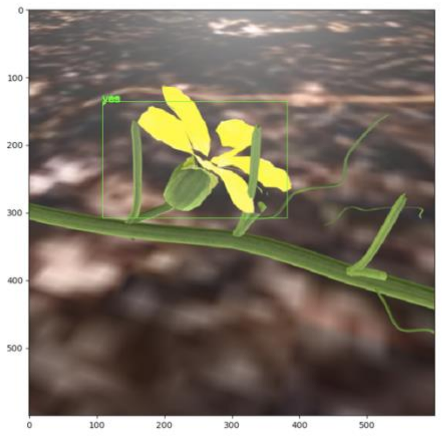
5. Results

After having implemented an already existing version of the network , which can be found in this git repository , I proceeded with creating different batch sizes of datasets containing flower images. Then I set up a new work environment in order to have access to a GPU in the CVLab.
After that , I used SSHFS to mount remote file systems over SSH , so that I can make changes on the fly and treat my droplet as local storage.
Everything was set in place , and I moved all my code from my local drive to the remote one . But then I had many problems trying to generate the datasets on the remote machine (the code starts running but doesn't show any output) , and when I tried copying the dataset directly to start training it , I had permission problems where I couldn't copy all the dataset and some images just wouldn't copy .
I wanted to do the training from my computer but it doesn't have a GPU and it was taking a long time to train one model with a small sized batch. Hence , In the following part I couldn't provide results for large datasets , but only a small one that was trained on my computer .

First of all , I started with a very small batch containing 20 images for training and 4 images for validation .
I trained it with ResNet 18 , 50 and 152
The results were not that good , after calculating the Intersection-Over-union which gives an average of : 0.353
Then I created a bigger dataset with 400 images of 1 particular species : the cantaloupe flower .
I was able to train it with ResNet18 and ResNet152 and it took more than 5 hours , but the results were better than the small batch of 24 pictures. For the prediction part it also didn't take much time for each image . Here are some of the results :



Trained with ResNet18



Trained with ResNet152

6. Future work

Due to time constraints there was a lot of things left out that could have been done to strengthen the results of this study such as more testing of different objects, distances and input sizes.

One thing that would be interesting is to train our models on bigger datasets and batches with different sizes . The first thing that should be done , in my opinion , is to generate datasets with different flower types having non-identical poses in the image . This will help train the model better and have more accurate results .

Another noteworthy experiment would be training the model with lower resolutions or zoomed in images and see if it could improve accuracy for smaller objects.
Moreover , another thing that was left out in this project was looking at the impact that lightning has on a models ability to detect objects, this is something that could be explored in future work.

References

[1] *Real time object detection on a Raspberry Pi* [Online].
Available : https://www.diva-portal.org/smash/get/diva2:1361039/FULLTEXT01.pdf

[2] *Visual Object Detection using Convolutional Neural Networks in a Virtual Environment* , by *Andreas Norrstig*
Available : https://www.diva-portal.org/smash/get/diva2:1307568/FULLTEXT01.pdf

[3] *5 Significant Object Detection Challenges and Solutions*(website).

Available : https://towardsdatascience.com/5-significant-object-detection-challenges-and-solutions-924cb09de9dd

[4] *Deep Neural Network Based Real-time Kiwi Fruit Flower Detection in an Orchard Environment*

Available : https://arxiv.org/abs/2006.04343

[5] *An Evaluation of Deep Learning Methods for Small Object Detection*

Available : https://www.hindawi.com/journals/jece/2020/3189691/

[6] *Implementing YOLO using ResNet as Feature extractor*

Available :https://medium.com/@m.khan/implementing-yolo-using-resnet-as-feature-extractor-5857f9da5014

[7] *Convolutional Neural Networks Backbones for Object Detection*

Available : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7340949/

[8] *Intersection over Union (IoU) for object detection*

Available :https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[9] *Faster RCNN Fine-Tune Implementation in Pytorch.*

Available : https://github.com/oke-aditya/pytorch_fasterrcnn

[10] *How To Use SSHFS to Mount Remote File Systems Over SSH*

Available :https://www.digitalocean.com/community/tutorials/how-to-use-sshfs-to-mount-remote-file-systems-over-ssh

[11] *TORCHVISION OBJECT DETECTION FINETUNING TUTORIAL*

Available : https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html