Final Documentation

Tomato Leaf Disease Detection using YOLO Algorithm

Member's Name: **Jethro P. Moleño**

Course/Section: CPE126-4/B1

Group No: 2

Acknowledgement

I want to express my deep appreciation to my instructor for his guidance and support throughout this research project. I also want to thank my colleagues for their assistance in model analysis; their diligent efforts have greatly enriched the depth and reliability of my findings. Furthermore, I am grateful to my mentors for their constructive criticism and valuable suggestions, which have played a key role in refining my manuscript and improving its clarity and rigor. Lastly, I extend my heartfelt gratitude to my family and loved ones for their patience, understanding, and encouragement throughout this journey. Their unwavering support has been an invaluable source of strength and motivation.

Abstract

Tomatoes stand out as a prevalent crop extensively employed in cooking and extensively cultivated by farmers. Yet, challenges persist in their cultivation, notably concerning leaf diseases that can impede their growth. This project aims to address such challenges through the implementation of a tomato leaf disease detection model employing the YOLOv5 algorithm. The dataset utilized, sourced from Kaggle, comprises 737 images of tomato leaves. The obtained results demonstrate the model's effectiveness, achieving a mean Average Precision (mAP) score of 76.07%. This underscores its potential utility in the tomato farming sector.

Introduction

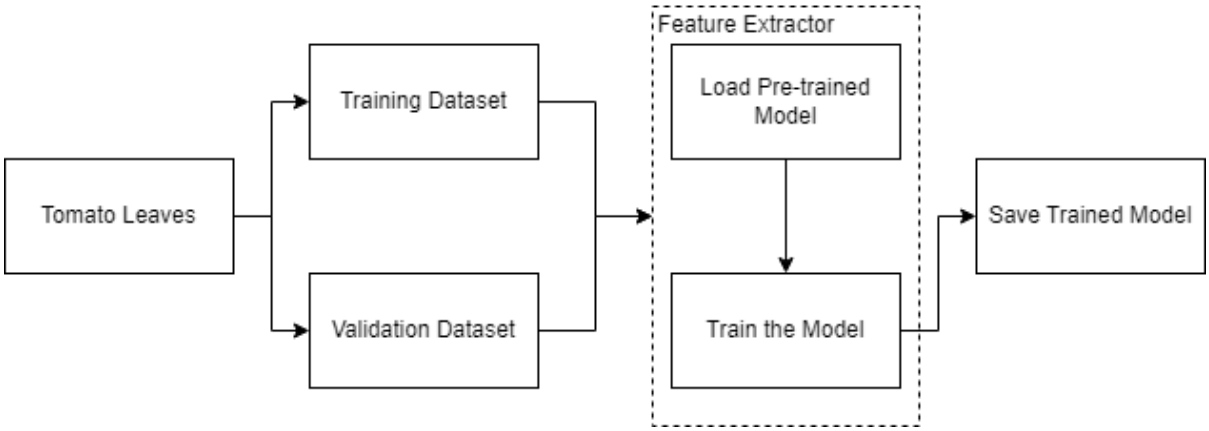
Tomatoes, scientifically referred to as Solanum lycopersicum, flourish in well-draining soil, making them a favored choice among farmers, with approximately 9 out of 10 opting to cultivate them in their fields. Moreover, numerous gardeners prefer to grow tomatoes in their plots to relish the freshness they impart to their culinary creations. Nonetheless, both farmers and gardeners may encounter hurdles in ensuring optimal plant growth. Challenges such as inadequate fruit development or the emergence of unattractive black spots resembling disease can manifest. The identification of diseases in tomato plants typically entails scrutinizing affected areas, noting any discoloration or lesions, and inspecting for signs of pests. It is advisable to avoid planting tomatoes continuously in the same soil for more than three years, along with similar vegetables like potatoes or eggplants, to mitigate soil depletion. To preserve soil fertility, it is recommended to precede tomato planting with a crop from the grass family, such as wheat, corn, rice, or sugarcane.

The objectives for this project are stated below:

- Develop a machine learning model utilizing YOLO to identify leaf diseases on tomato plants.
- Utilize the Tomato Leaf dataset from Kaggle.
- Evaluate the proposed project using statistical treatment metrics.

Methods and Results

The methods used are based on the block diagram below. The researcher followed the YOLOv5 custom training google colab notebook from the Github of the official YOLO documentation. Firstly, the dataset were gathered using the Kaggle website which consists of 737 tomato leaf images. Additionally, the dataset was annotated in YOLOv5 Pytorch format. The pre-processing was already applied in each image, namely, resized to 640x640 resolution and auto oriented the pixel data. After that, the dataset was divided into 645 images for training, 61 images for validation and 31 images for testing the model.



After preparing the dataset, the process for training the model was done. Firstly, the environment as well as the dependencies were installed using the code below:

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml  # install

import torch
import utils
display = utils.notebook_init()  # checks
```

After running the code, the YOLOv5 version, Pytorch version and the Hardware accelerator type was shown as displayed below:

```
YOLOv5 🚀 v7.0-307-g920c721e Python-3.10.12 torch-2.2.1+cu121 CUDA:0
(Tesla T4, 15102MiB)
```

Furthermore, the GPU specifications that is used by the Google Collab was also shown by running the code `!nvidia-smi` *which displayed the following specifications:*

```
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   32C    P8              9W /  70W |      3MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

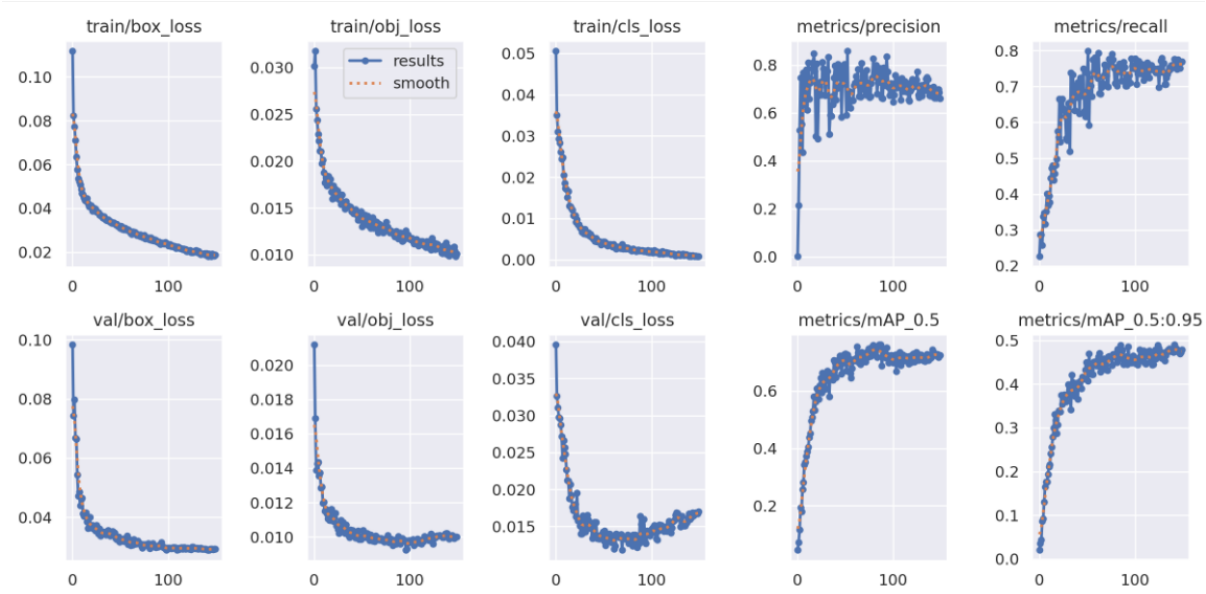
After that, the researcher mounted the Google Drive folder where the dataset from Kaggle was uploaded. Then the following code was ran to execute the training of the model.

```
!python train.py --img 640 --batch 16 --epochs 150 --data
/content/drive/MyDrive/MachineLearning/TomatoLeafDIseasesDetect.v1i.yol
ov5pytorch/data.yaml --weights yolov5s.pt --cache
```

In this line of code, the image size was set to 640x640, which must be the same size as the training images. The batch and epoch is default value from the tutorial, and the data path or the dataset location is from the google drive. The YOLO algorithm used was YOLOv5s and the images were

cached for faster training. The training time was completed after 15 minutes and the weights was saved in runs/train/exp3 path. Also, the statistical metrics were displayed in png format as well as the confusion matrix in the same filepath.

Based on the results gathered, the model performed well in the validation phase, with a mAP score of 76.07% as shown in the figure. The mAP score, which pertains to the overall mAP score of all the classes was shown after validation. This means that the model performed well and can be applied for other test images since mostly, good models should have around more than 70% mAP.



Based on the graphs shown above, the average precision was 77.6% and the average recall was 76%. This implies that the model was able to detect all instances of the class as well as assessing model s capability to quantify true positives among all positive prediction from the validation testing.

After the validation phase, the testing dataset was predicted using the trained model by entering the code below:

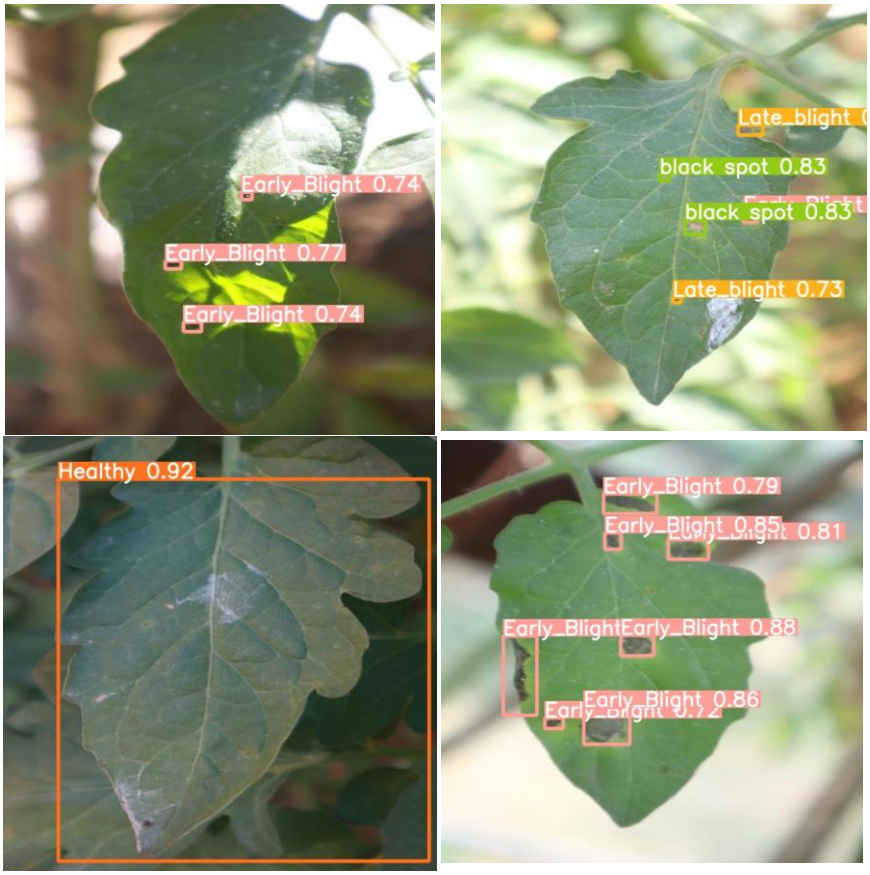
```
!python detect.py --weights runs/train/exp3/weights/best.pt --img 640 --conf 0.1 --source /content/drive/MyDrive/MachineLearning/TomatoLeafDIseasesDetect.v1i.yolov5pytorch/test/images
```

Following the model testing, the inference of all test images were shown by using the code below:

```
import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.jpg'):
#assuming JPG
    display(Image(filename=imageName))
    print("\n")
```

This line of code resulted on displaying all the test images with the bounding box of the classes and confidence score.



Comparing the trained model from other models on other researchers that used different model and dataset with the same tomato leaf disease detection models, it still performs well given with only 737 images compared to the dataset of other models which ranges around 3000 to 50000 images.

Conclusion

In the presented study, a model utilizing the YOLOv5 algorithm was developed by the researcher to identify various diseases affecting tomato leaves. The dataset sourced from Kaggle comprises seven distinct classes, including bacterial spot, early blight, healthy, late blight, leaf mold, target spot, and black spot. The researcher leveraged the YOLOv5 custom dataset training Google Colab file and adhered to its guidelines. Experimental findings revealed a mean Average Precision (mAP) score of 76.07% for the model. Notably, both precision and recall metrics demonstrated promising results, approximately 77.6% and 77% respectively, indicating the efficacy of the model. Future endeavors could focus on enhancing the model's performance by augmenting the dataset to further fine-tune it, a strategy employed in recent studies aimed at improving Tomato leaf disease detection.

References

Trivedi NK, Gautam V, Anand A, Aljahdali HM, Villar SG, Anand D, Goyal N, Kadry S. Early Detection and Classification of Tomato Leaf Disease Using High-Performance Deep Neural Network. Sensors (Basel). 2021 Nov 30;21(23):7987. doi: 10.3390/s21237987. PMID: 34886991; PMCID: PMC8659659.

Mohit Agarwal, Abhishek Singh, Siddhartha Arjaria, Amit Sinha, Suneet Gupta, ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network, Procedia Computer Science,Volume 167, 2020, Pages 293-301, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.03.225

Tools used:

https://www.kaggle.com/datasets/farukalam/tomato-leaf-diseases-detection-computer-vision/data

https://colab.research.google.com

https://drive.google.com/

https://universe.roboflow.com

https://colab.research.google.com/github/roboflow-ai/yolov5-custom-training-tutorial/blob/main/yolov5-custom-training.ipynb#scrollTo=1jS9_BxdBBHL

https://docs.ultralytics.com/guides/yolo-performance-metrics/

Appendix

Source code:

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml  # install
import torch
import utils
display = utils.notebook_init()  # checks
!nvidia-smi
!python train.py --img 640 --batch 16 --epochs 150 --data
/content/drive/MyDrive/MachineLearning/TomatoLeafDIseasesDetect.v1i.yol
ov5pytorch/data.yaml --weights yolov5s.pt --cache
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
!python detect.py --weights runs/train/exp3/weights/best.pt --img 640 -
-conf 0.1 --source
```

```
/content/drive/MyDrive/MachineLearning/TomatoLeafDIseasesDetect.v1i.yol
ov5pytorch/test/images
#display inference on ALL test images
import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.jpg'):
#assuming JPG
    display(Image(filename=imageName))
    print("\n")
```