Configuration Manual

MSc Research Project
MSc. in Data Analytics

Mardwin Alejandro Cardenas Rodríguez
Student ID: 20144237

School of Computing
National College of Ireland

Supervisor: Majid Latifi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Mardwin Alejandro Cardenas Rodriguez
Student ID:	20144237
Programme:	MSc. in Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Majid Latifi
Submission Due Date:	31/01/2022
Project Title:	Configuration Manual
Word Count:	963
Page Count:	15

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	31st January 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Mardwin Alejandro Cardenas Rodríguez
20144237

1 Introduction

This setup documentation contains all of the relevant data, including the equipment I utilized, software and hardware specifications, crucial code screenshots, and reproducibility requirements. The specifications are detailed in Section 2, which includes the Software Standard and Hardware Specifications.

2 Specifications

The following chapters go through the software and hardware requirements for this proposed study.

2.1 Software Configurations

As we can see in the Table 1, the summarize of the software configurations that it have been employed over this investigation. Figure 2 demonstrates the hardware and operating system performance.

Table 1: Software

Software	Configuration
Operational System	Windows 10 Home Single Language
Online IDE	Google Colab notebooks
Coding Language	Python
Coding Language Version	Python 3.7
Additional Tools Used	RoboFlow, LabelMe, Google Colab

2.2 Hardware Configurations

Table 2 illustrates the hardware configurations used in this investigation.

Table 2: Hardware

Hardware	Configuration
System	Intel(R) Core(TM) 4210U
Operation System	Windows 10 Home Single Language
RAM	6.00 GB
Hard Disk	Acer Aspire V3- EA107501
Libraries	cv2, os, torch, roboflow, yaml, utils
Graphic Card	Intel (R) HD Graphics Family

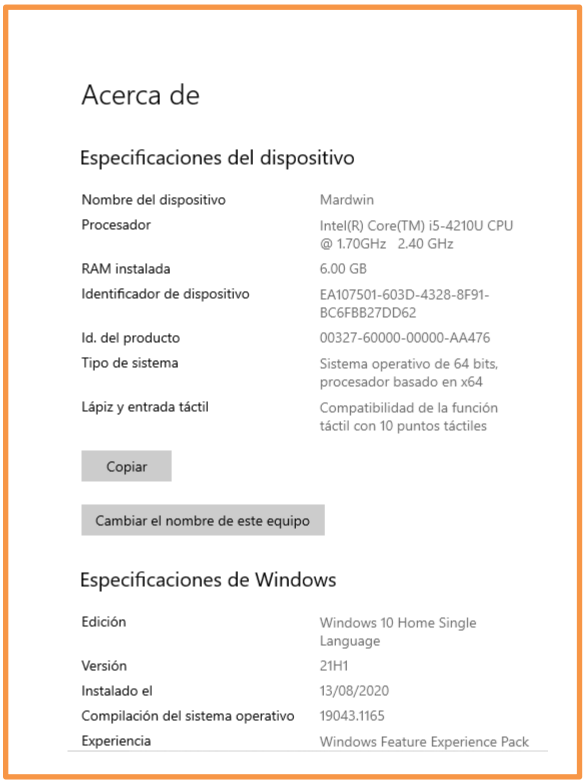


Figure 1: Device and Windows Specifications

3 Integrated Development Environment

To conduct the research and run the code, Google Colab's Jupyter Notebook was utilized. Anaconda Navigator is now loaded, and the Figure 2 bellow shows how it appears once I start it.

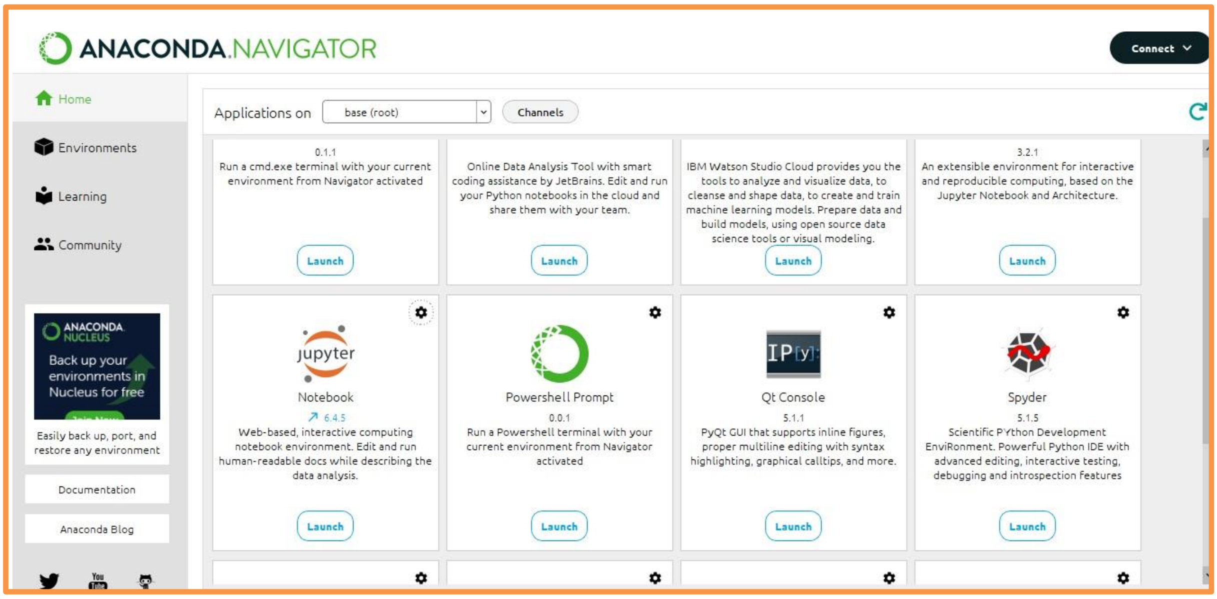


Figure 2: Anaconda Navigator

To utilize the Jupyter Notebook, launch the Anaconda Navigator and select on jupyter notebook, as can be seen in Figure 3. You should see something like this.

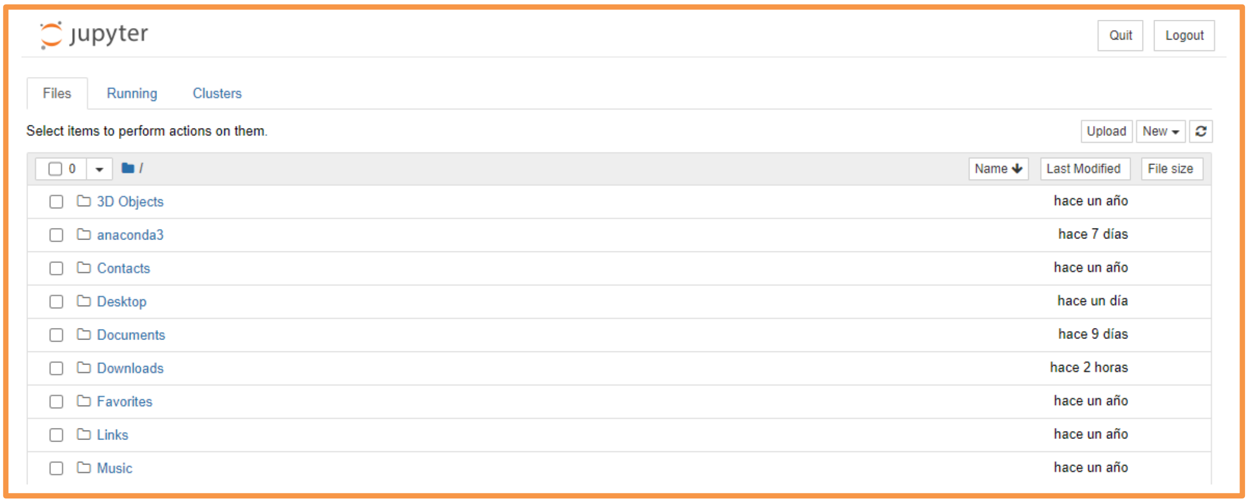


Figure 3: Jupyter notebook launched

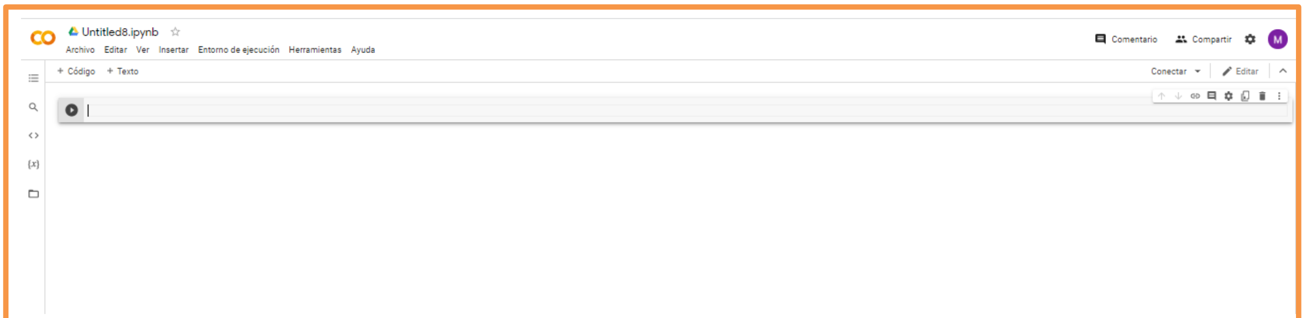


Figure 4: Google Colab launched

4 LabelMe

After Anaconda is installed, In an Anaconda Prompt we must run the following commands:

4.1 Download Labelme

```
# python3
conda create --name=labelme python=3.6
conda activate labelme
pip install labelme
```

Figure 5: Installation commands

1.1 Open Labelme

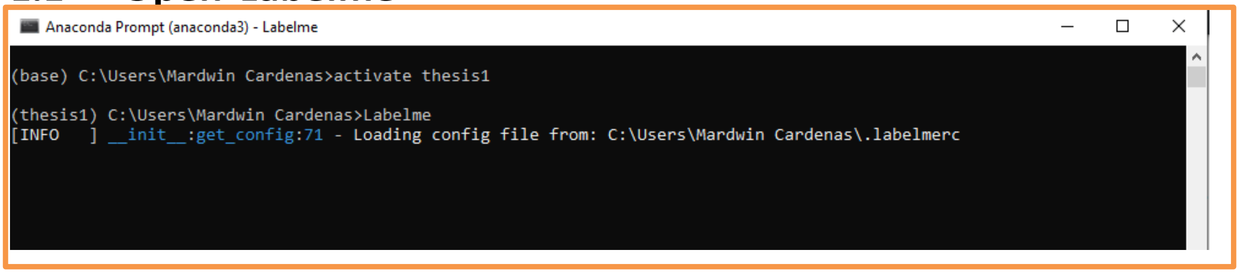


Figure 6: Anaconda Prompt command

Figure 7: LabelMe Application

1.1 Select Directory to open and save images

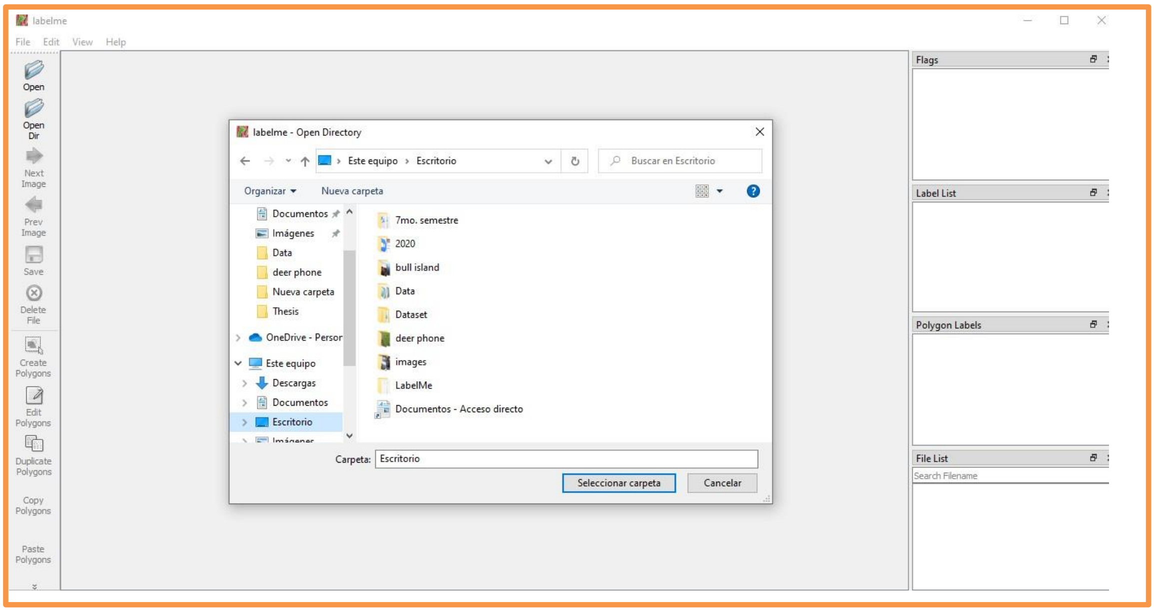


Figure 8: LabelMe Directory Selection

1.2 Draw and Save Annotations

When we select and save, the photograph is stored to the place we have specified previously. Afterwards we just click on the next photograph and repeat the process for each picture..

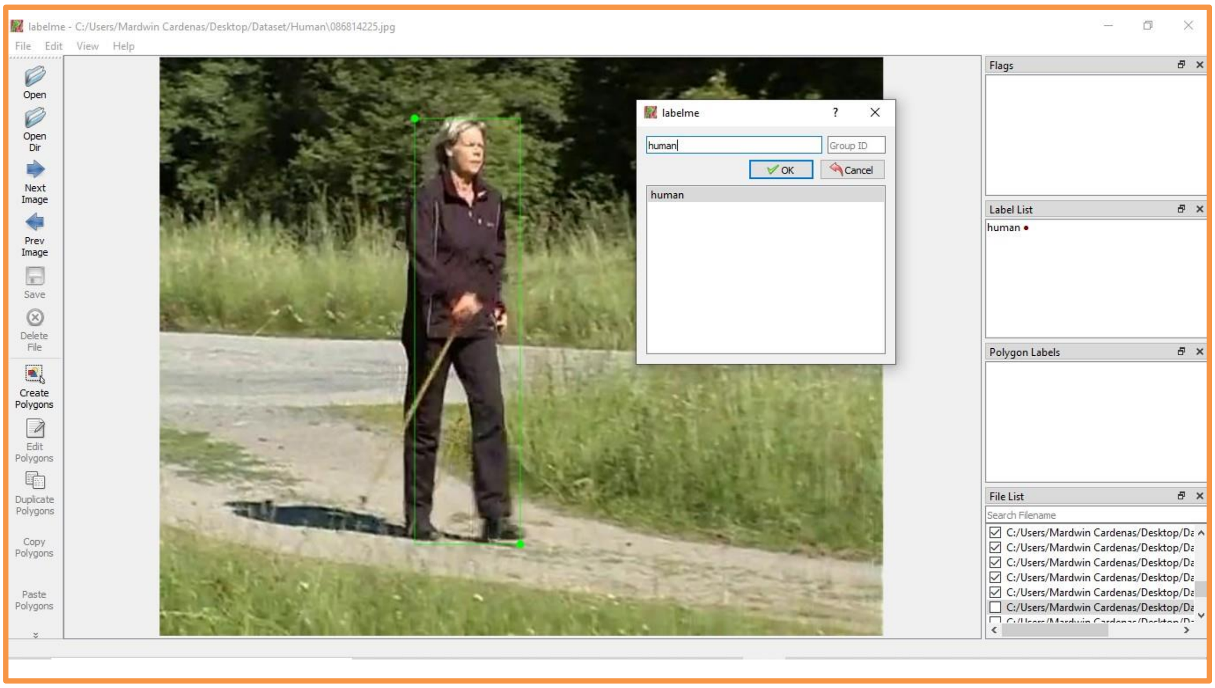
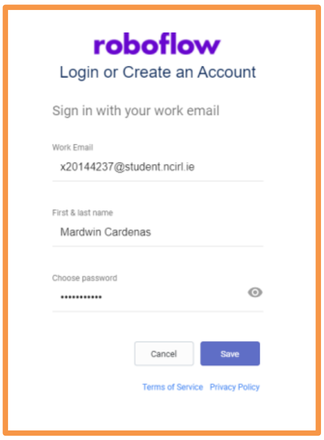


Figure 9: Image labeling

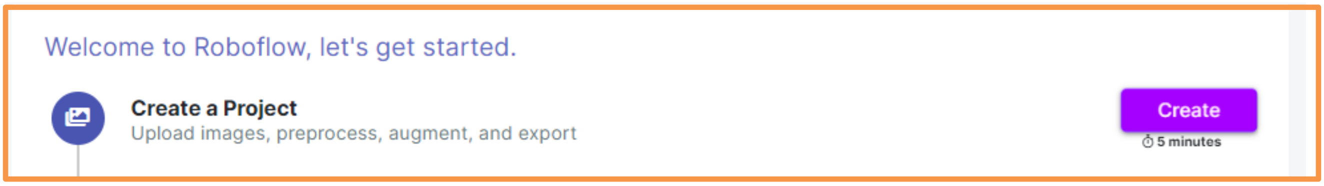
2 RoboFlow

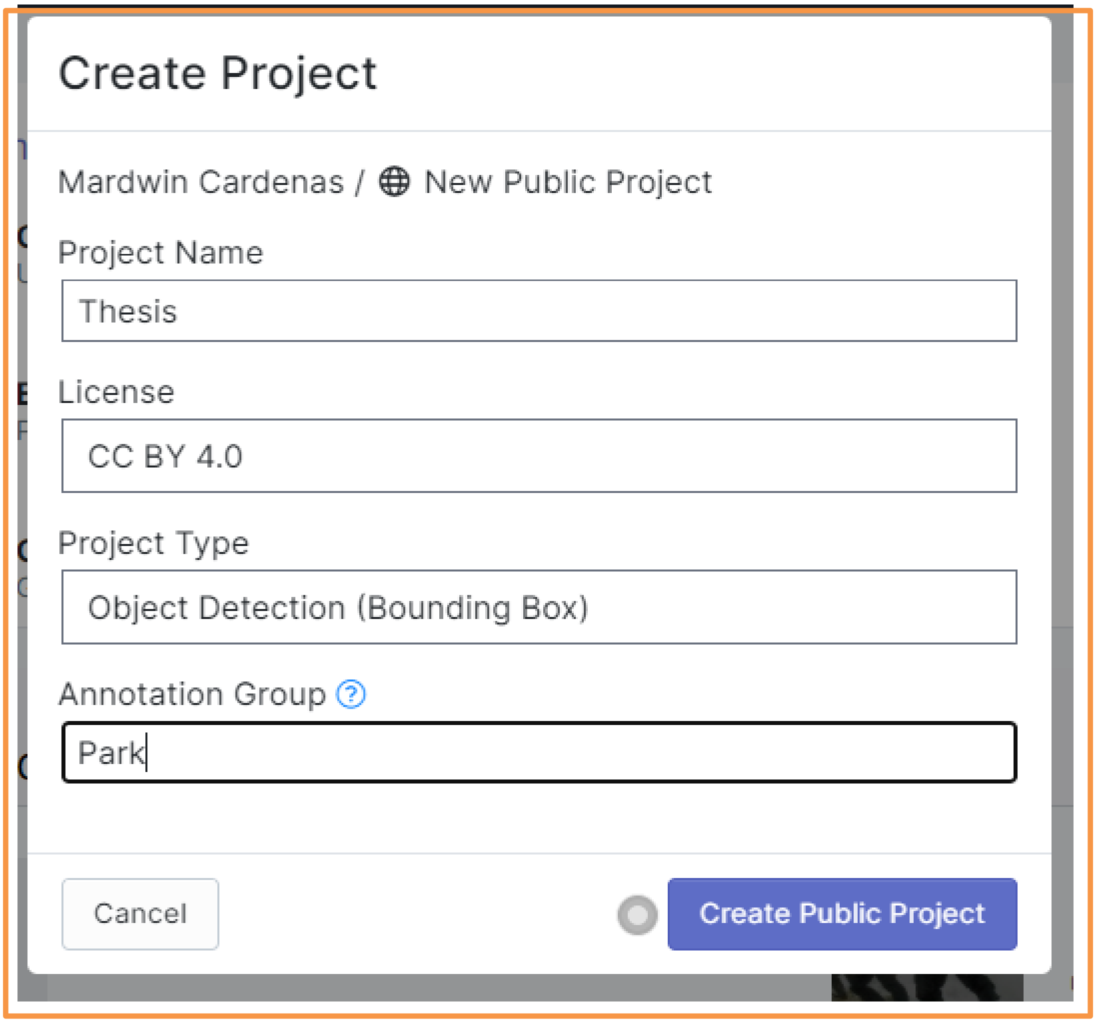
Roboflow intends to make computer vision more accessible to the general public by simplifying and streamlining machine learning procedure. This application can allow us to focus on the real challenge instead of worrying of annotation formats, building scripts to process the raw photographs, or understanding how to code.

2.1 Create an Account

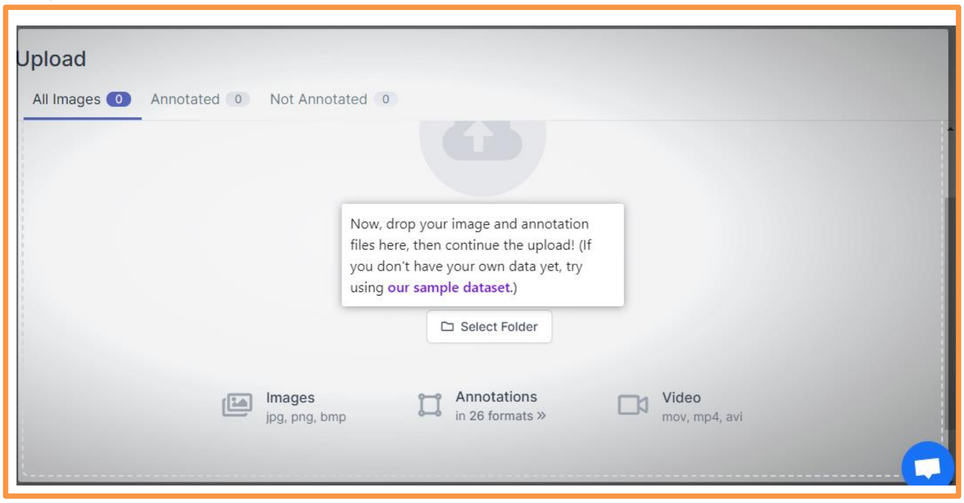


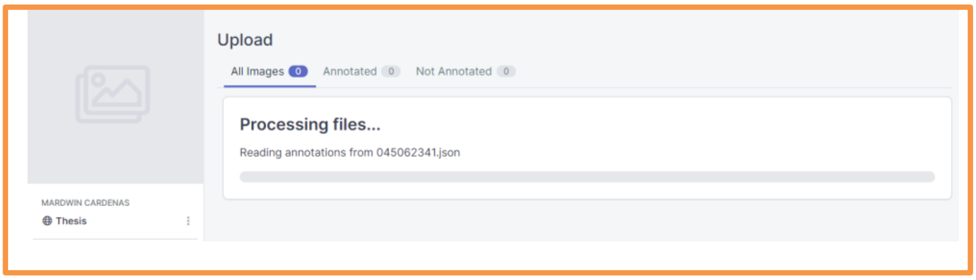
2.2 Create a Project



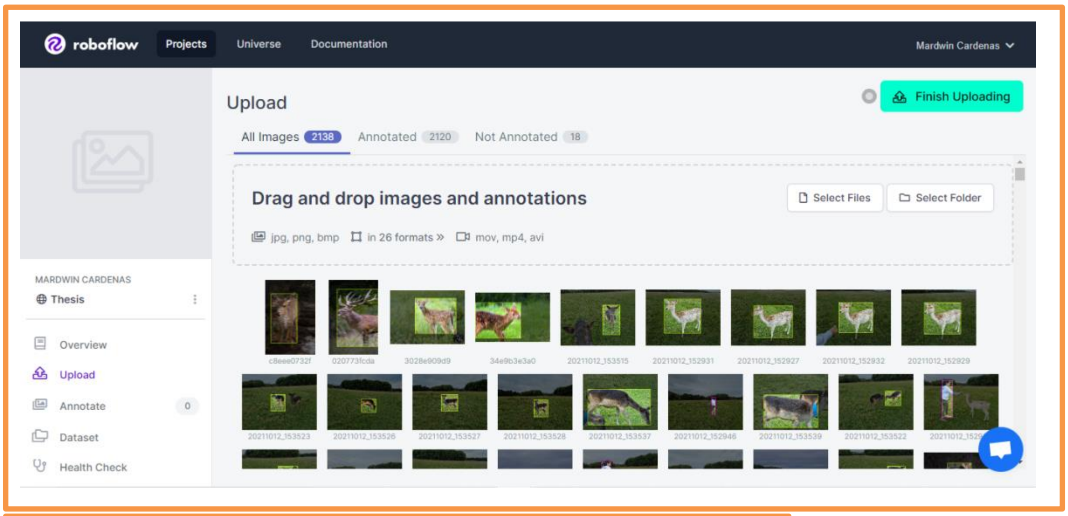


2.3 Upload



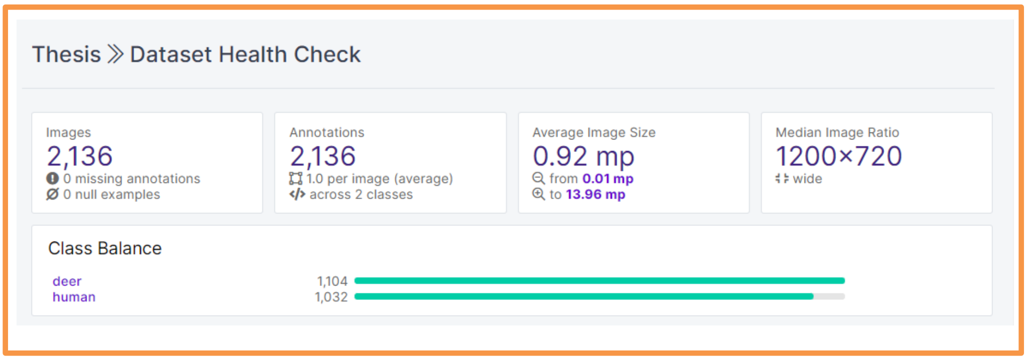


2.4 Image Preprocessing

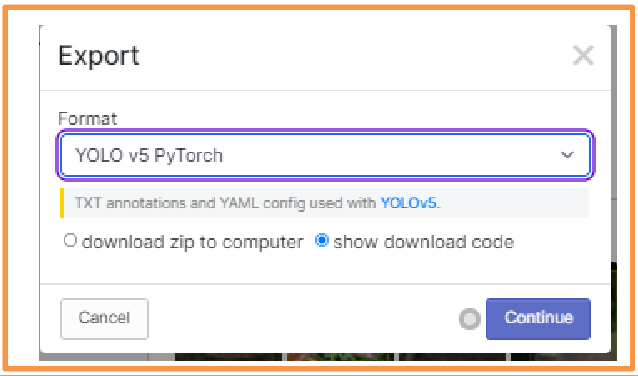


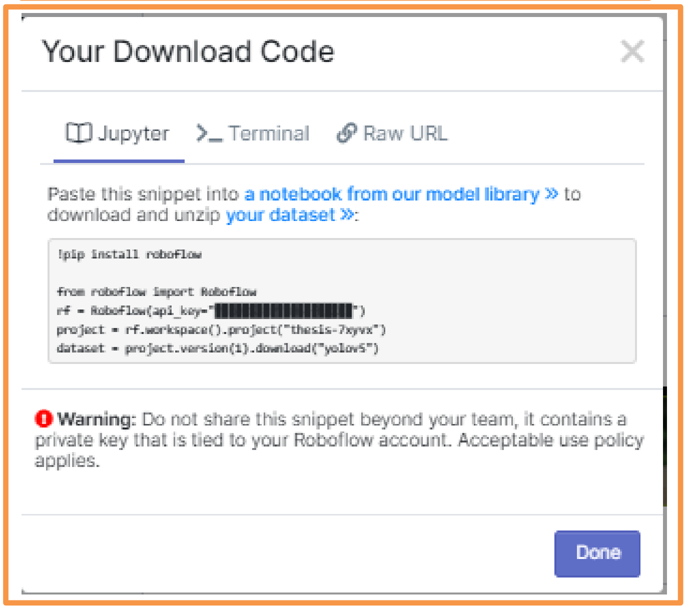
Source Images	**Images:** 2,136 **Classes:** 2 **Unannotated:** 0
Train/Test Split	**Training Set:** 1.5k images **Validation Set:** 427 images **Testing Set:** 214 images
Preprocessing	**Auto-Orient:** Applied **Resize:** Stretch to 416×416
Augmentation	**90° Rotate:** Clockwise, Counter-Clockwise **Grayscale:** Apply to 25% of images

2.5 Data Health check



2.6 Data Export





3 YOLOv5 Training Phase

3.1 Installing the YOLOv5 Environment

To get started with YOLOv5, we'll download the repository and apply the requirements. This will prepare our development platform so that object identification training and interpretation instructions can be executed.

```
# clone YOLOv5 repository
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
!git reset --hard 886f1c03d839575afecb059accf74296fad395b6

Cloning into 'yolov5'...
remote: Enumerating objects: 10142, done.
remote: Total 10142 (delta 0), reused 0 (delta 0), pack-reused 10142
Receiving objects: 100% (10142/10142), 10.43 MiB | 8.12 MiB/s, done.
Resolving deltas: 100% (7031/7031), done.
/content/yolov5
HEAD is now at 886f1c0 DDP after autoanchor reorder (#2421)
```

Figure 9: Clone form Repository

After that, we can look at our Google Colab setup and begin installing requirements..

```
[ ] # install dependencies as necessary
    !pip install -qr requirements.txt  # install dependencies (ignore errors)
    import torch

    from IPython.display import Image, clear_output  # to display images
    from utils.google_utils import gdrive_download  # to download models/datasets

    # clear_output()
    print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else 'CPU'))

       |████████████████████████████████| 596 kB 4.4 MB/s
    Setup complete. Using torch 1.10.0+cu111 CPU
```

Figure 10: Dependencies Installation

We will be able to reduce training time by using the GPU, Torch is preconfigured on Colab, and it is a beneficial characteristic. As we do not attempt to run this code in local, there are not additional setups to be follow for YOLOv5.

3.2 Download Correctly Formatted Custom Dataset

Loading into this notebook our data

```
#follow the link below to get your download code from from Roboflow
!pip install roboflow

from roboflow import Roboflowimport torch
from IPyt

Requirement already satisfied: roboflow in /usr/local/lib/python3.7/dist-packages (0.2.0)
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.7/dist-packages (from roboflow) (0.19.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from roboflow) (2.8.2)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.7/dist-packages (from roboflow) (6.0)
Requirement already satisfied: kiwisolver==1.3.1 in /usr/local/lib/python3.7/dist-packages (from roboflow) (1.3.1)
Requirement already satisfied: chardet==4.0.0 in /usr/local/lib/python3.7/dist-packages (from roboflow) (4.0.0)
Requirement already satisfied: idna==2.10 in /usr/local/lib/python3.7/dist-packages (from roboflow) (2.10)
Requirement already satisfied: opencv-python>=4.1.2 in /usr/local/lib/python3.7/dist-packages (from roboflow) (4.1.2.30)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.7/dist-packages (from roboflow) (7.1.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from roboflow) (1.15.0)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.7/dist-packages (from roboflow) (4.62.3)
Requirement already satisfied: cycler==0.10.0 in /usr/local/lib/python3.7/dist-packages (from roboflow) (0.10.0)
Requirement already satisfied: pyparsing==2.4.7 in /usr/local/lib/python3.7/dist-packages (from roboflow) (2.4.7)
Requirement already satisfied: urllib3==1.26.6 in /usr/local/lib/python3.7/dist-packages (from roboflow) (1.26.6)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from roboflow) (3.2.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.7/dist-packages (from roboflow) (1.19.5)
Requirement already satisfied: certifi==2021.5.30 in /usr/local/lib/python3.7/dist-packages (from roboflow) (2021.5.30)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from roboflow) (2.26.0)
Requirement already satisfied: wget in /usr/local/lib/python3.7/dist-packages (from roboflow) (3.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.7/dist-packages (from requests->roboflow) (2.0.8)
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in Thesis-try-3 to yolov5pytorch: 100% [147630036 / 147630036] bytes
Extracting Dataset Version Zip to Thesis-try-3 in yolov5pytorch:: 100%|██████████| 8878/8878 [00:10<00:00, 832.49it/s]
```

Figure 11:

3.3 Model Architecture

We can use the pre created yaml file because it specifies our model's characteristics, such as the number of classes, anchors, and layers.

```
# this is the YAML file Roboflow wrote for us that we're loading into this notebook with our data
%cat {dataset.location}/data.yaml

names:
- deer
- human
nc: 2
train: Thesis-try-3/train/images
val: Thesis-try-3/valid/images
```

Figure 12: Data Model Specifications

3.4 Train the Model

Based on the information we have. Yolov5s is now ready to train with our yaml files on hand. To start training, execute the training instruction with the following parameters:

Img:	416 x416
Size of the batch	16
Epoch	100
Data location	{dataset.location}/data.yaml
Cfg	./models/custom_yolov5s.yaml
Weights	' '
Name	yolov5_result

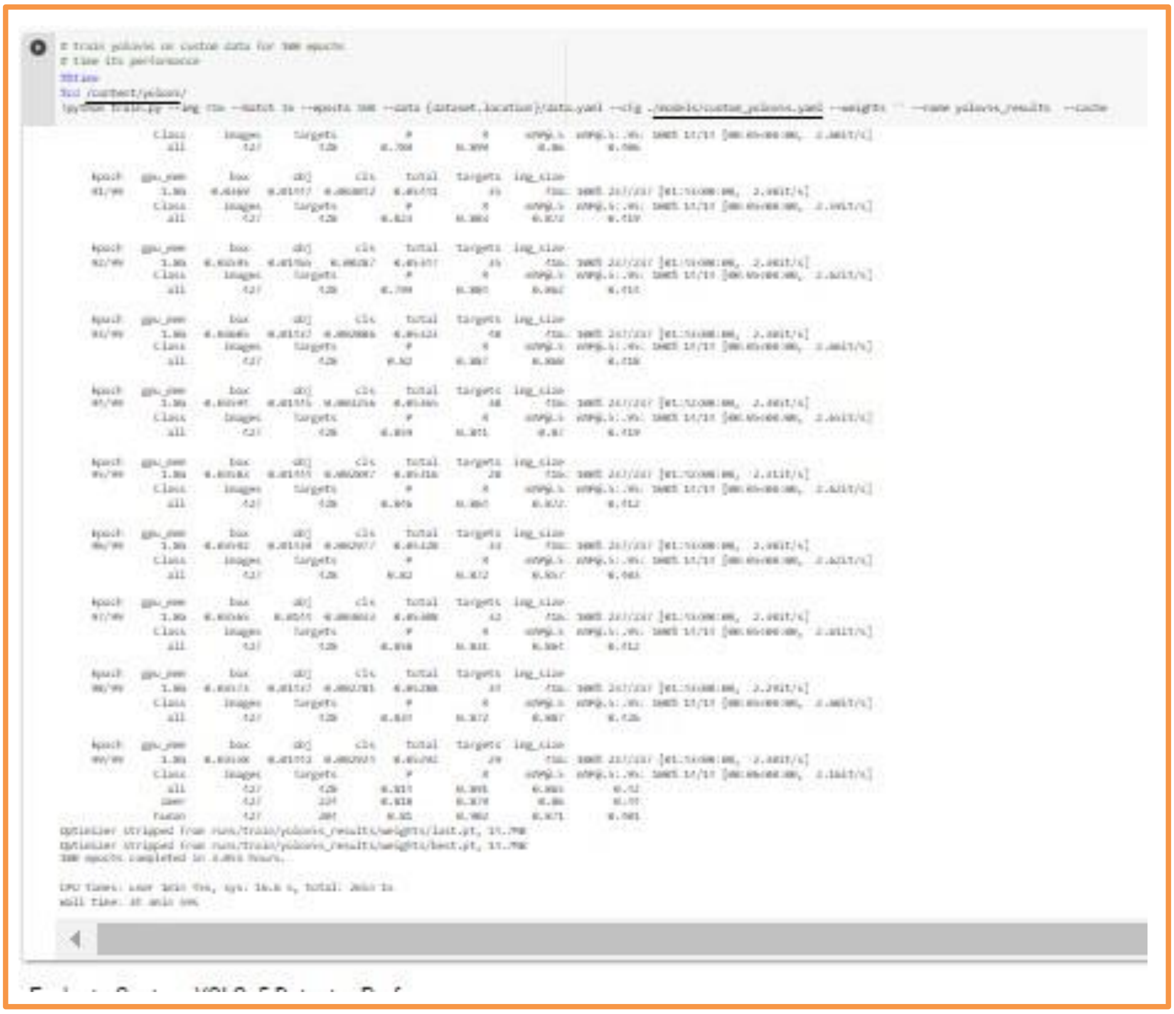
And run the training command:

```python
# define number of classes based on YAML
import yaml
with open(dataset.location + "/data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])
```

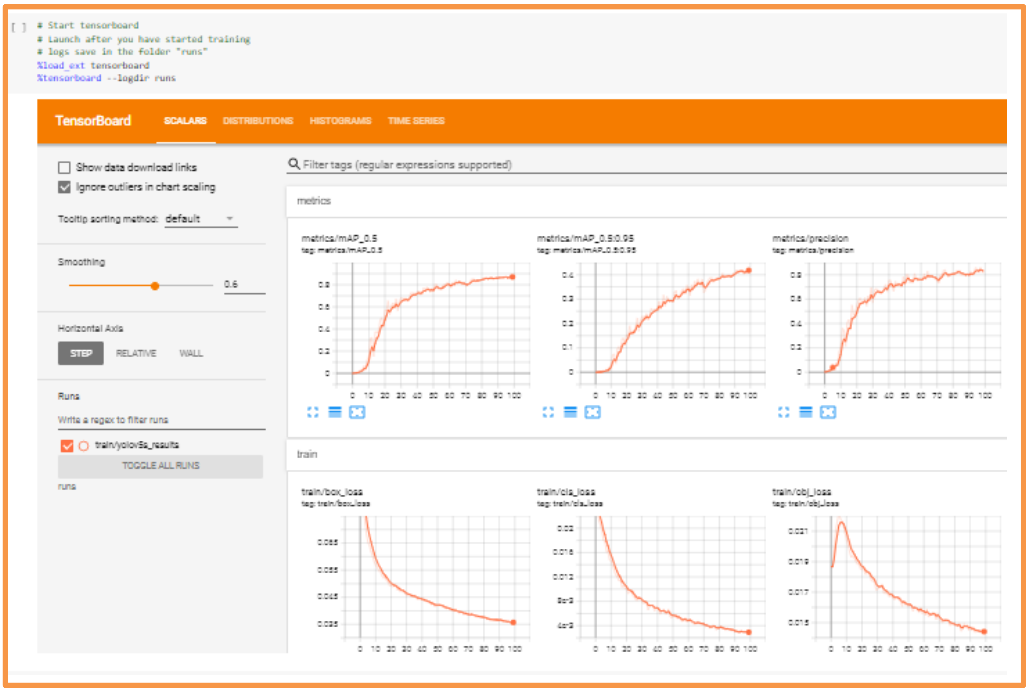
```python
#this is the model configuration we will use for our tutorial
%cat /content/yolov5/models/yolov5s.yaml
```

```yaml
# parameters
nc: 80  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, C3, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, C3, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```

```python
#customize iPython
from IPython.core.magic import register_line_cell_magic

@register_line_cell_magic
def writetemplate(line, cell):
    with open(line, 'w') as f:
        f.write(cell.format(**globals()))
```



3.5 Evaluate Custom YOLOv5 Detector Performance



3.6 Run Inference with Trained Weights

```
[ ] # trained weights are saved by default in our weights folder
    %ls runs/

    detect/  train/

[ ] %ls runs/train/yolov5s_results/weights

    best.pt  last.pt

[ ] # when we ran this, we saw .007 second inference time. That is 140 FPS on a TESLA P100!
    # use the best weights!
    %cd /content/yolov5/
    !python detect.py --weights runs/train/yolov5s_results/weights/best.pt --img 416 --conf 0.4 --source /content/gdrive/MyDrive/deer

    #display inference on ALL test images
    import glob
    from IPython.display import Image, display

    for imageName in glob.glob('/content/yolov5/runs/detect/exp12/*.mp4'): #assuming JPG
        display(Image(filename=imageName))
        print("\n")
```

3.7 Export Trained Weights for Future Inference

```
from google.colab import drive
drive.mount('/content/gdrive')

Mounted at /content/gdrive
```

```
%cp /content/yolov5/weights/last_yolov5s_custom.pt /content/gdrive/My\ Drive
```

4 References

Ali, S. M. (n.d.). Comparative analysis of yolov3, yolov4 and yolov5 for sign language detection, p. 2021.
URL: www.ijariie.com2393

EVOLUTION OF YOLO ALGORITHM AND YOLOV5: THE STATE-OF-THE-ART OBJECT DETECTION ALGORITHM (n.d.).
URL: https://www.semanticscholar.org/paper/Do-Thuan-EVOLUTION-OF-YOLO-ALGORITHM-AND-YOLOV5%3A-OF-Thuan/68d608f3c45014b1b74660a178ab190147310d9e