ENEL 645 (FALL 2021)
DATA MINING & MACHINE LEARNING

PROJECT REPORT

OBJECT DETECTION IN IMAGES WITH MACHINE LEARNING

17 December 2021

GROUP NO	MEMBER 1	MEMBER 2	DATASET NO	ALGORITHM
2	SYED AHMAD SOHAIL	MARIA JAN	2	2

UNIVERSITY OF CALGARY

ABSTRACT

Object detection is used for locating instances of objects in pictures and videos using computer vision technique. Object detection algorithm usually use machine learning and deep learning models to give specified results. When humans look at the object, intellect matches the recognition of the instance to compare and know what the object is, the computer vision uses the same way to compare with the model that has been trained on similar images to recognise the object. Humans have been living with the conflict of animal where animals stray close to humans in search of food and often come to prey on the crops or cattle and sometimes humans' causalities are also reported. In Alberta bear seasons are also very common and animal detection can be used to know what animal and where has been detected to preserve life of humans and animal both. This project focuses on animal detection for which YOLOv5 was used.

INTRODUCTION

With image processing, object detection is one of the critical ongoing research in the domain of computer vision. Numerous applications are arising in this new world of technologies where object detection plays a vital role. Everything viewed through the lens of robots. Moving object detection is one of the current research, which is being made better for more problems to be solved, catering to Artificial Intelligence.[1] One of the major use case of this application, security has become a vital problem around the globe with increased terror and crime. Surveillance, detection and tracking have turned out to be essential, for the kind of places that requires security. Another application of this system is to detect a human in the frame and notify back the stakeholder of this system. A project that will be deployed by us and will be working on the concept of computer vision to detect animals.

Methodology

STEPS FOLLOWED

DATASET USED:

https://www.kaggle.com/antoreepjana/animals-detection-images-dataset/code

ALGORITHM USED: YOLOv5

YOLOv5 is a group of compound-scaled object detection models prepared on the COCO dataset, and incorporates simple use for Test Time Augmentation (TTA), model ensembling, hyperparameter advancement, and product to ONNX, CoreML and TFLite [4].

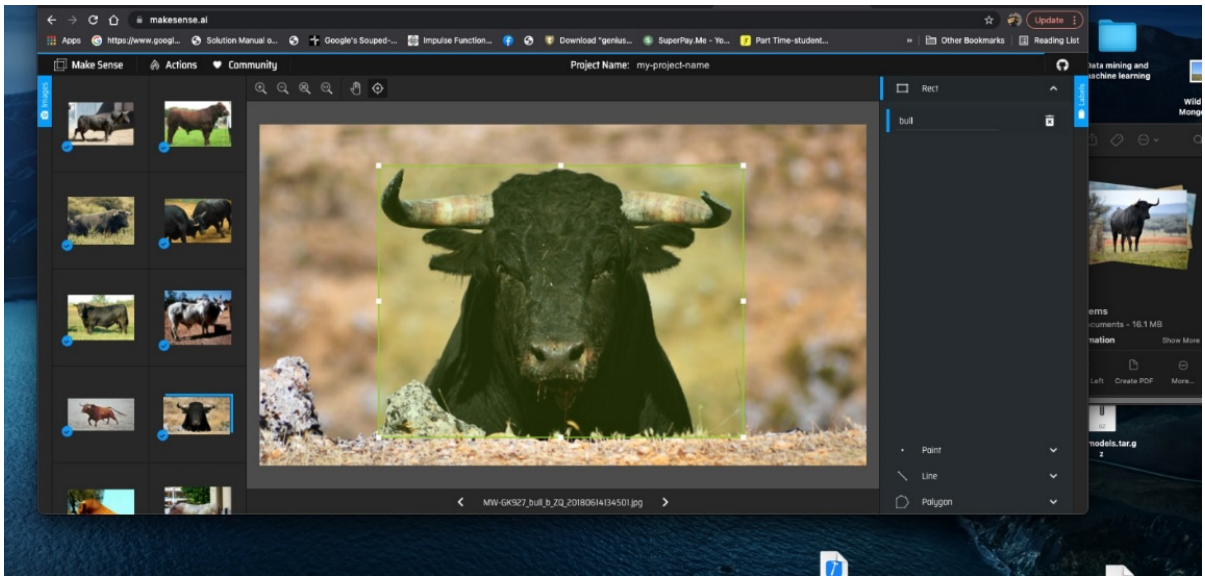
There are basically three layers of Yolov5:

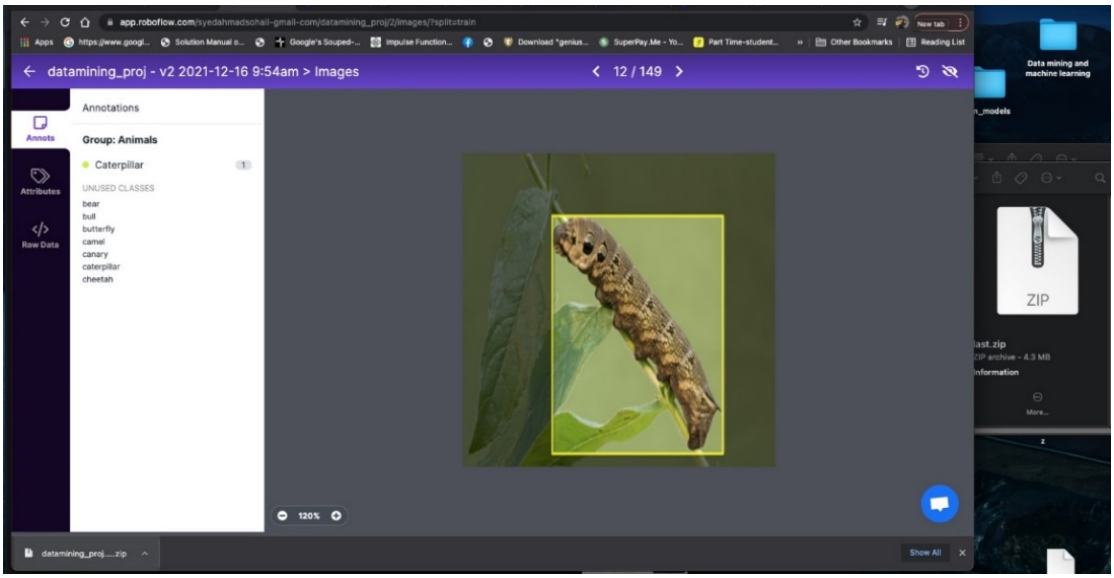
1. Backbone: CSPDarknet.
   Model Backbone is mainly used to extract important features from the given input image.
2. Neck: PANet
   Model Neck is mainly used to generate feature pyramids. Feature pyramids help models to generalized well on object scaling. It helps to identify the same object with different sizes and scales.
3. Head: Yolo Layer
   The model Head is mainly used to perform the final detection part. It applied anchor boxes on features and generates final output vectors with class probabilities, objectness scores, and bounding boxes.

<u>FOR BOUNDING BOXES AND LABELLING:</u>

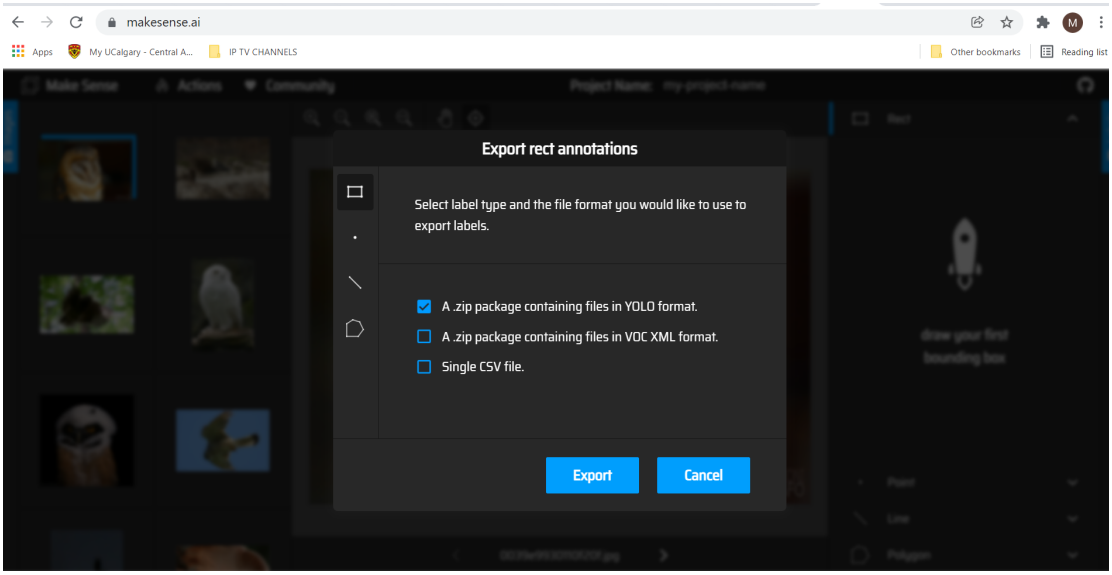
Step1: Using https://www.makesense.ai/, uploaded the dataset folder for each animal category

Step2: Created bounding boxes and labelled the animals accordingly



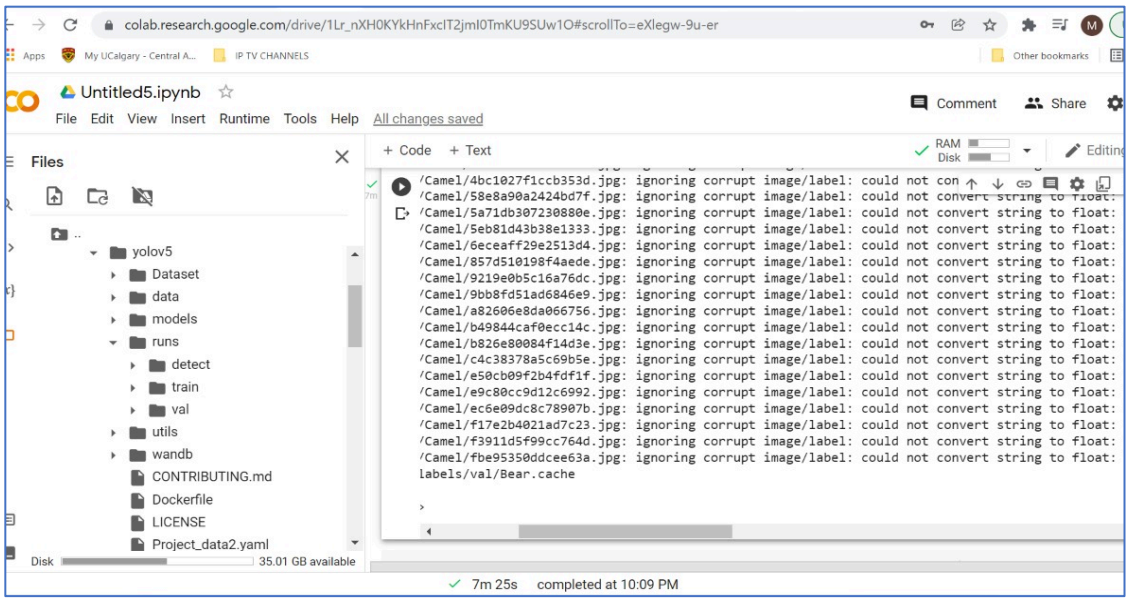


Step3: Exported the labelled annotations in Yolo Format



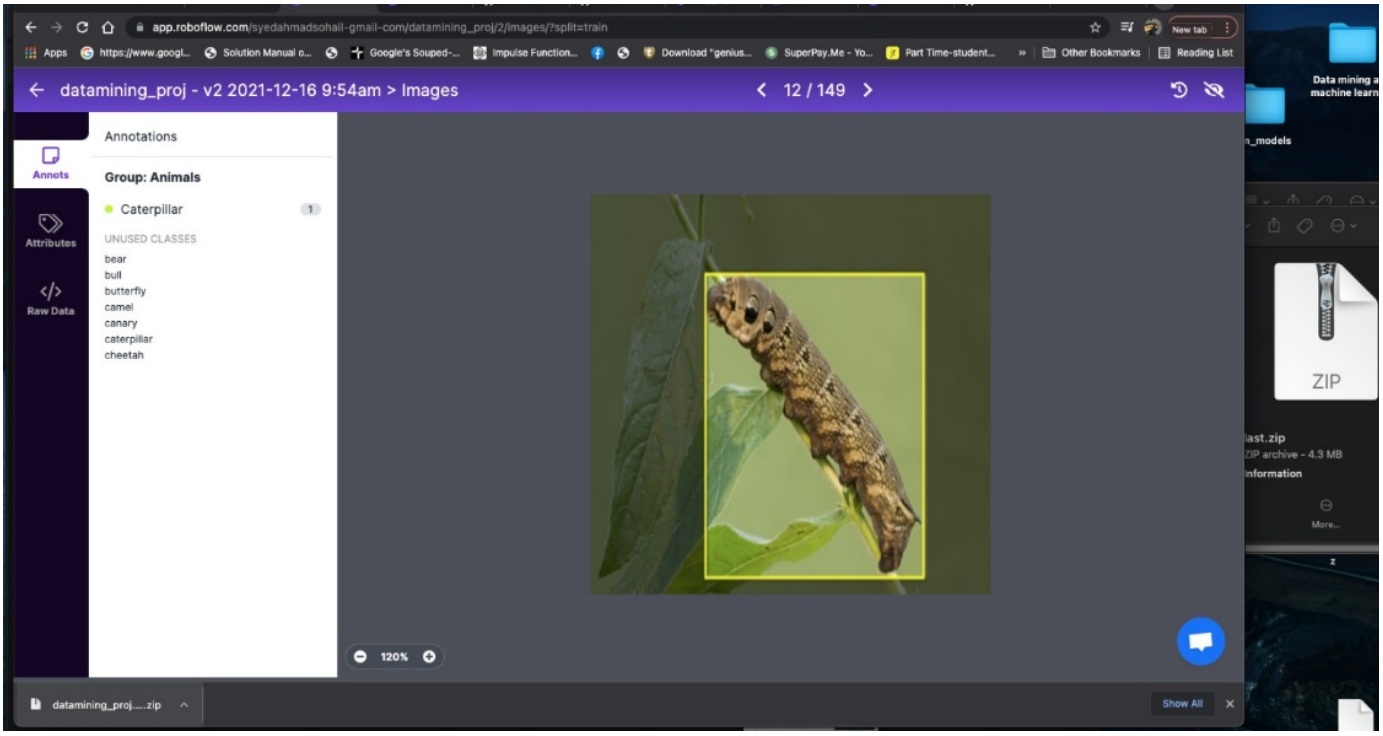
<u>LIMITATIONS:</u>

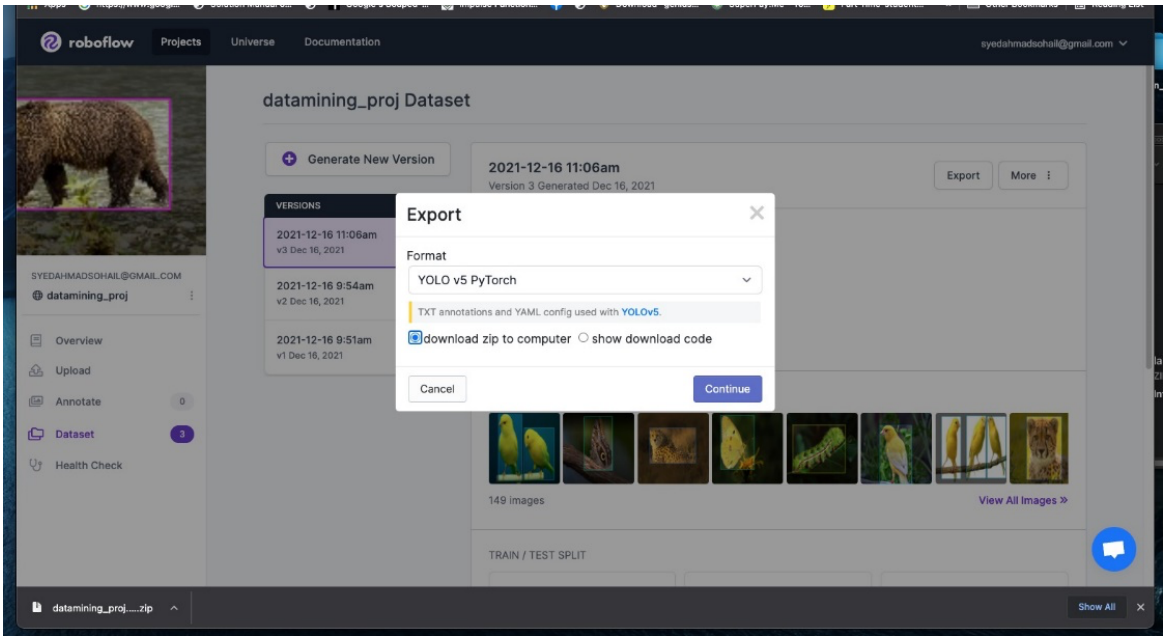
Results in binary format, training labels and images did not sync, and showed corrupt labels/ images while training for training on makesense.ai



USING ROBOFLOW:

- Created bounding boxes and labelled the animals accordingly
- Exported the labelled annotations in Yolo Format





GOOGLE COLAB:

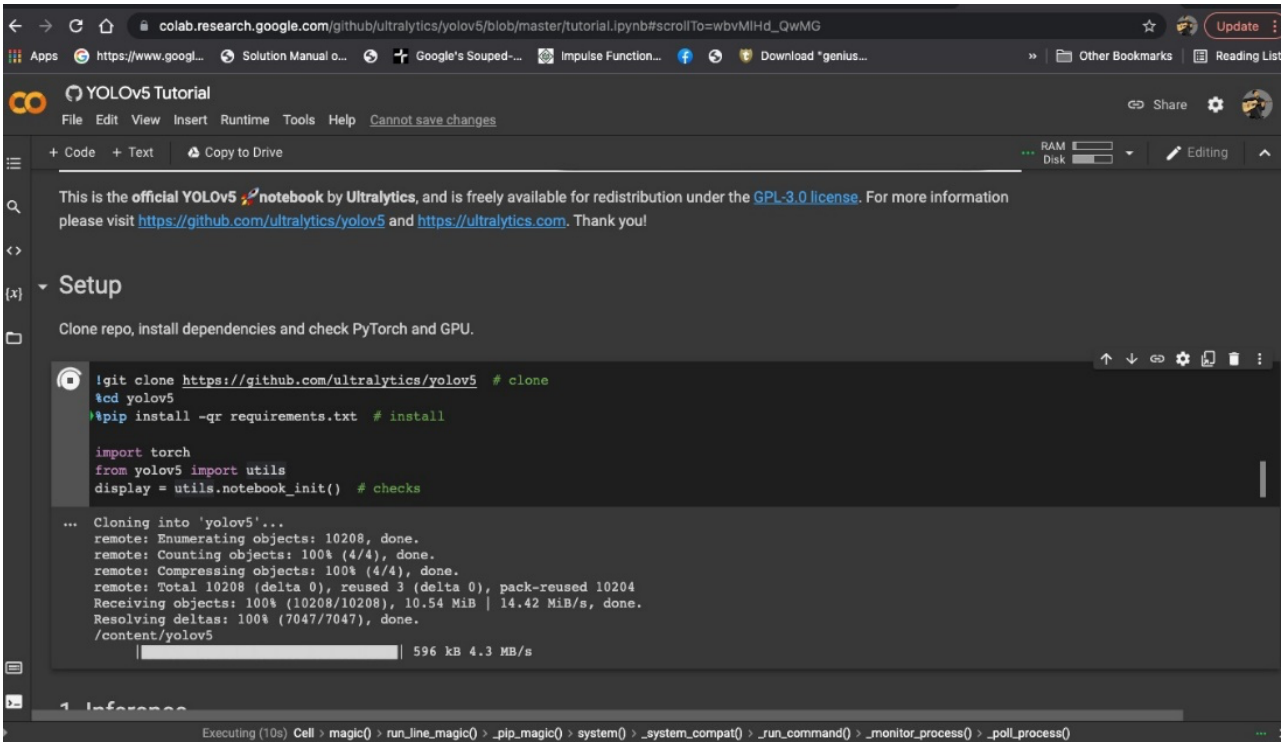
NVIDIA GPU: Tesla K80 GPU (12GB), by changing 'runtime type' to GPU

```
!nvidia-smi

Thu Dec 16 07:49:43 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| 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 K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   34C    P8    26W / 149W |      3MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

SETTING UP ENVIRONMENT:

Step 1: On Google Colab, mount drive

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

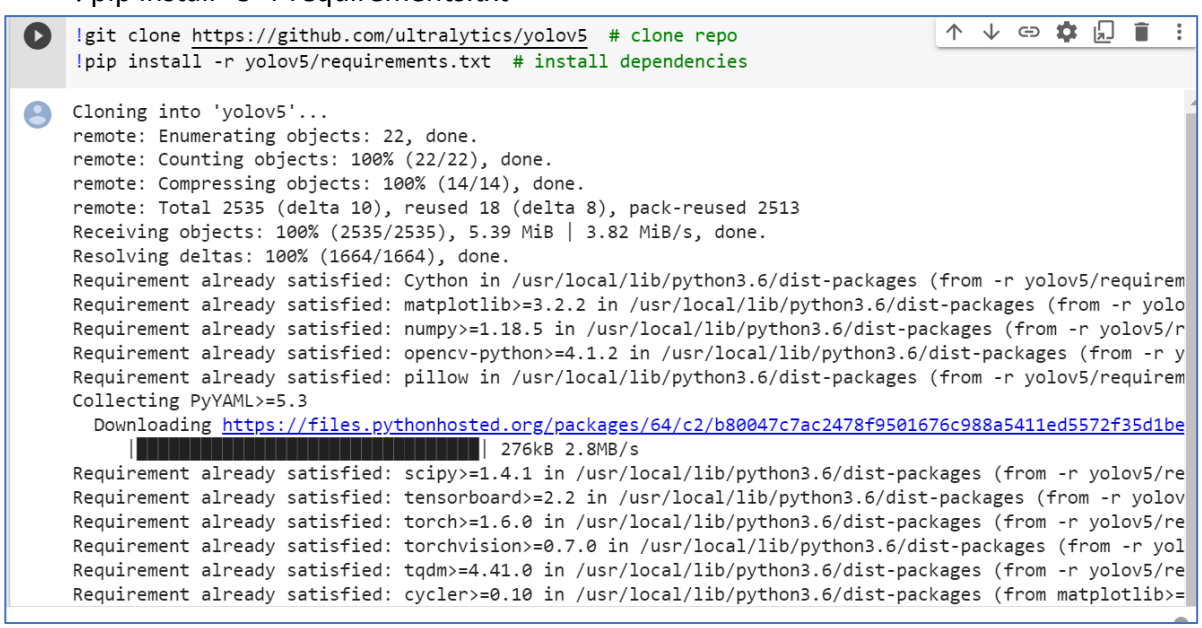


Step2: Cloning the git repo of Yolov5 Algorithm

! git clone https://github.com/ultralytics/yolov5

Step3: Installing dependencies

! pip install -U -r requirements.txt



```
!git clone https://github.com/ultralytics/yolov5  # clone repo
!pip install -r yolov5/requirements.txt  # install dependencies
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 2535 (delta 10), reused 18 (delta 8), pack-reused 2513
Receiving objects: 100% (2535/2535), 5.39 MiB | 3.82 MiB/s, done.
Resolving deltas: 100% (1664/1664), done.
Requirement already satisfied: Cython in /usr/local/lib/python3.6/dist-packages (from -r yolov5/requirem
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.6/dist-packages (from -r yolo
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.6/dist-packages (from -r yolov5/r
Requirement already satisfied: opencv-python>=4.1.2 in /usr/local/lib/python3.6/dist-packages (from -r y
Requirement already satisfied: pillow in /usr/local/lib/python3.6/dist-packages (from -r yolov5/requirem
Collecting PyYAML>=5.3
  Downloading https://files.pythonhosted.org/packages/64/c2/b80047c7ac2478f9501676c988a5411ed5572f35d1be
     |▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓| 276kB 2.8MB/s
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.6/dist-packages (from -r yolov5/re
Requirement already satisfied: tensorboard>=2.2 in /usr/local/lib/python3.6/dist-packages (from -r yolov
Requirement already satisfied: torch>=1.6.0 in /usr/local/lib/python3.6/dist-packages (from -r yolov5/re
Requirement already satisfied: torchvision>=0.7.0 in /usr/local/lib/python3.6/dist-packages (from -r yol
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.6/dist-packages (from -r yolov5/re
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=
```

Step4: For Setup

```
import torch
from yolov5 import utils
display = utils.notebook_init()  # checks
```

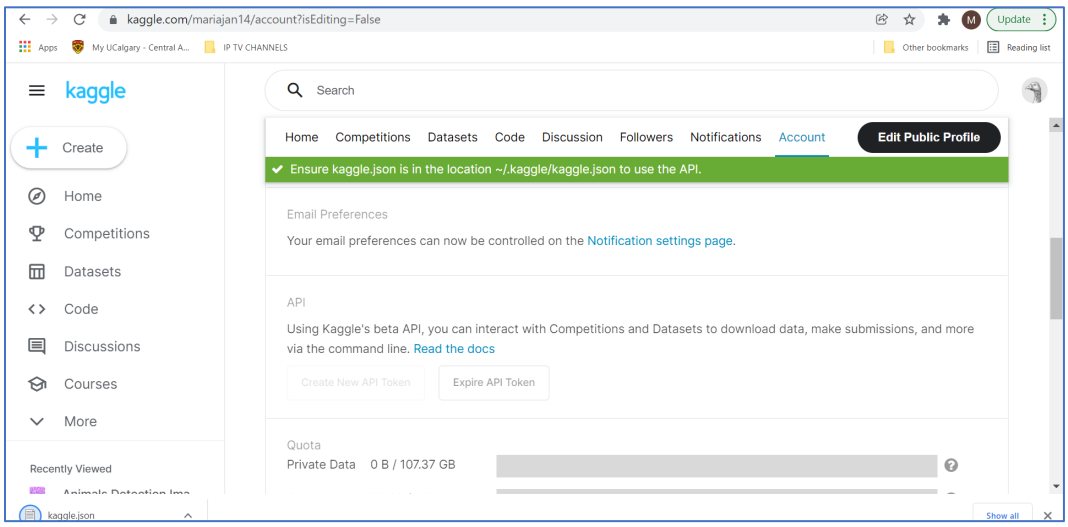
FOR KAGGLE DATASET:

Step5: Downloading Kaggle datasets via Kaggle API

Get the API key from your account

Visit www.kaggle.com ⇨ login ⇨ My Account ⇨ Create New API Token

The "kaggle.json" file will be auto downloaded.



Step6: Upload the kaggle.json file, using following code snippets in Google Colab

```
from google.colab import files
files.upload()                    #to upload the kaggle.json from PC
```

Step7: To create the folder path, and to upload the kaggle.json in the required path

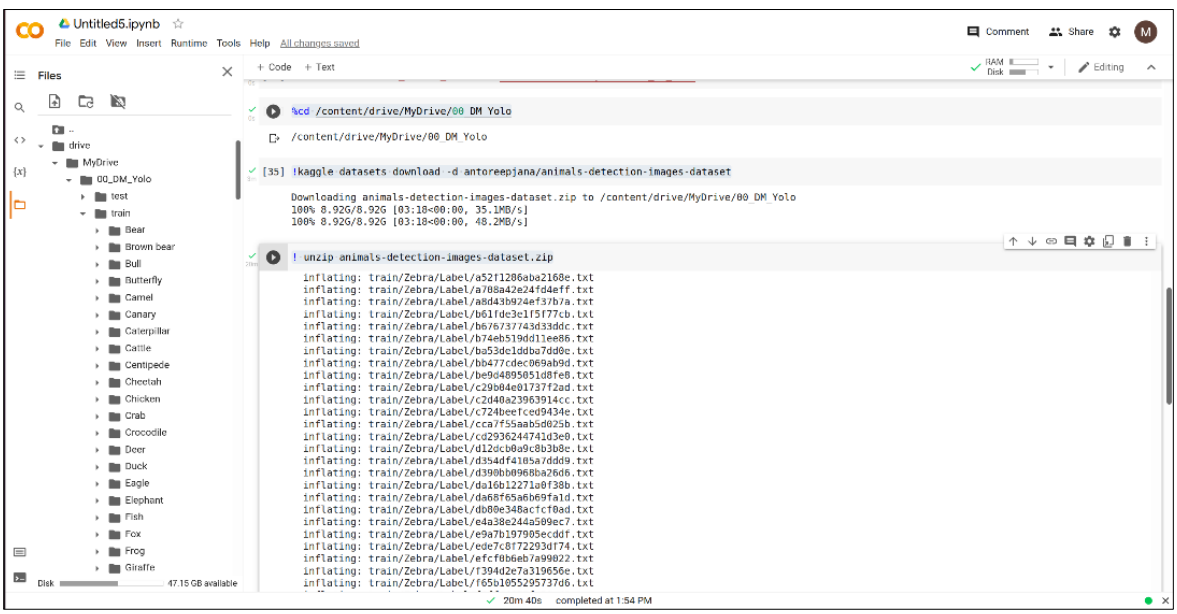
```
! pip install -q kaggle
! mkdir -p ~/.kaggle
! cp kaggle.json ~/.kaggle/
! ls ~/.kaggle
! chmod 600 /root/.kaggle/kaggle.json  # set permission
```

Step8: Download the required dataset from kaggle website using API Command

```
! kaggle datasets download -d antoreepjana/animals-detection-images-dataset
```

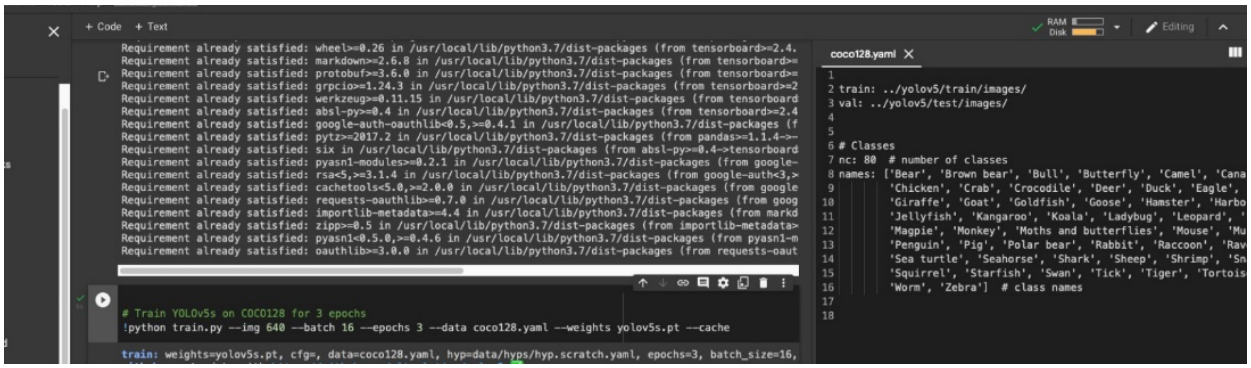
Step9: Unzip the dataset folder

! unzip animals-detection-images-dataset.zip



Step 10: Arranging the Dataset of Images and Labels

Step 11: Creating YOLOv5 model configuration file



TRAIN YOLOV5S:

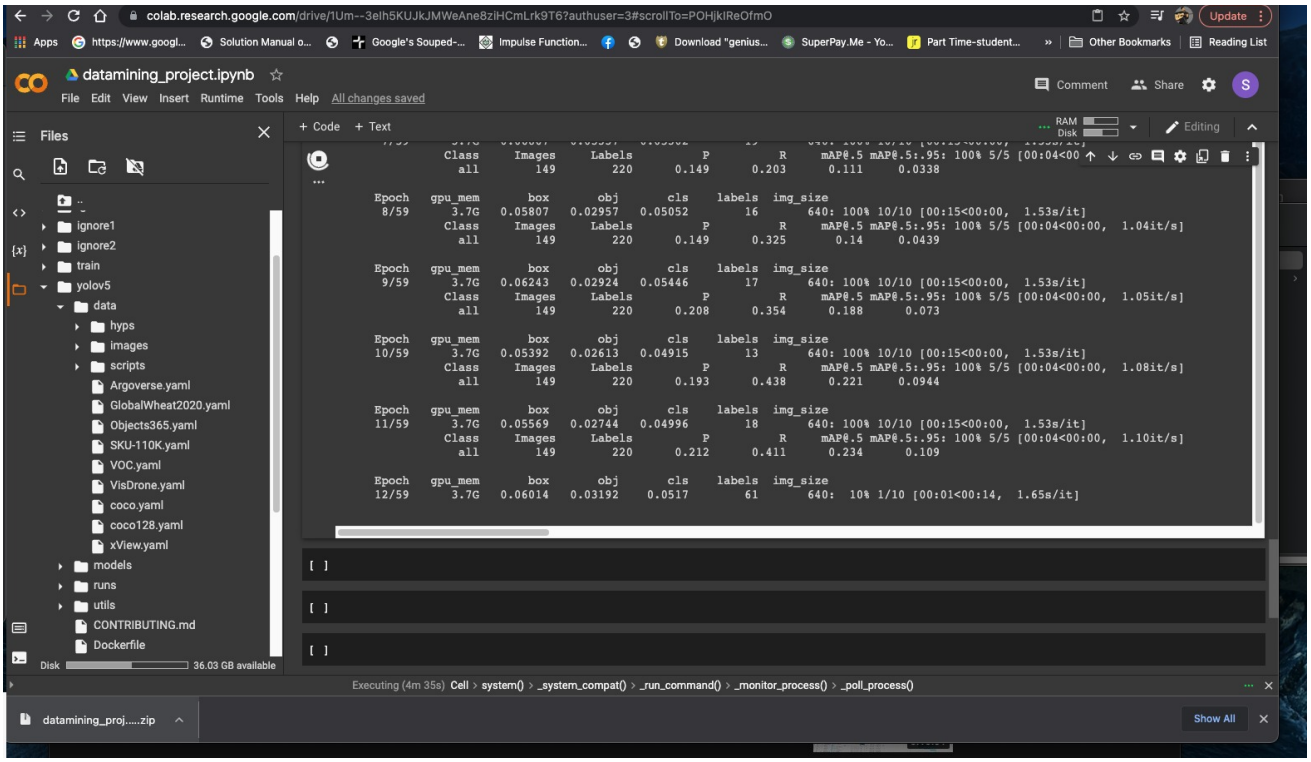
! python train.py --img 640 --batch 2 --epochs 60 --data coco128.yaml --weights yolov5s.pt

```
                 from  n    params  module                                  ar
  0                -1  1      3520  models.common.Conv                      [3
  1                -1  1     18560  models.common.Conv                      [3
  2                -1  1     18816  models.common.C3                        [6
  3                -1  1     73984  models.common.Conv                      [6
  4                -1  2    115712  models.common.C3                        [1
  5                -1  1    295424  models.common.Conv                      [1
  6                -1  3    625152  models.common.C3                        [2
  7                -1  1   1180672  models.common.Conv                      [2
  8                -1  1   1182720  models.common.C3                        [5
  9                -1  1    656896  models.common.SPPF                      [5
 10                -1  1    131584  models.common.Conv                      [5
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [N
 12           [-1, 6]  1         0  models.common.Concat                    [1
 13                -1  1    361984  models.common.C3                        [5
 14                -1  1     33024  models.common.Conv                      [2
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [N
 16           [-1, 4]  1         0  models.common.Concat                    [1
 17                -1  1     90880  models.common.C3                        [2
 18                -1  1    147712  models.common.Conv                      [1
 19          [-1, 14]  1         0  models.common.Concat                    [1
 20                -1  1    296448  models.common.C3                        [2
 21                -1  1    590336  models.common.Conv                      [2
 22          [-1, 10]  1         0  models.common.Concat                    [1
 23                -1  1   1182720  models.common.C3                        [5
 24      [17, 20, 23]  1     16182  models.yolo.Detect                      [1
Model Summary: 270 layers, 7022326 parameters, 7022326 gradients, 15.8 GFLOPs
```

Transferred 343/349 items from yolov5s.pt
Scaled weight_decay = 0.0005
optimizer: SGD with parameter groups 57 weight, 60 weight (no decay), 60 bias
albumentations: version 1.0.3 required by YOLOv5, but version 0.1.12 is curren train: Scanning '/content/bull_data/train/labels.cache' images and labels... 2 train: Caching images (0.0GB ram): 100% 24/24 [00:00<00:00, 32.84it/s]
val: Scanning '/content/bull_data/train/labels.cache' images and labels... 24

```
val: Caching images (0.0GB ram):  100%  24/24 [00:01<00:00,  23.97it/s] Plotting labels
to runs/train/exp8/labels.jpg...

AutoAnchor: 2.70  anchors/target,  1.000 Best Possible Recall (BPR). Current  anc
Image sizes 640   train, 640   val
Using 2  dataloader workers
Logging results  to runs/train/exp8
Starting training for 60   epochs...

Epoch     gpu_mem    box              obj         cls                labels             img_size
0/59       0.51G          0.08064          0.03389         0                6             640: 100%  1
Class      Images       Labels              P                 R         mAP@.5 mA
all          24                 30        0.00465                0.433            0.00361

Epoch     gpu_mem         box              obj             cls                labels             img_size
1/59              0.7G       0.083          0.03191         0                6             640: 100%  1
Class             Images       Labels              P                 R           mAP@.5 mA
all                24            30            0.00737         0.467            0.00504

Epoch     gpu_mem         box              obj             cls                labels             img_size
2/59              0.703G    0.08519       0.03368         0                9             640: 100%  1
Class             Images       Labels            P                     R                mAP@.5 mA
All                24             30                0.0107                0.533            0.00716
```

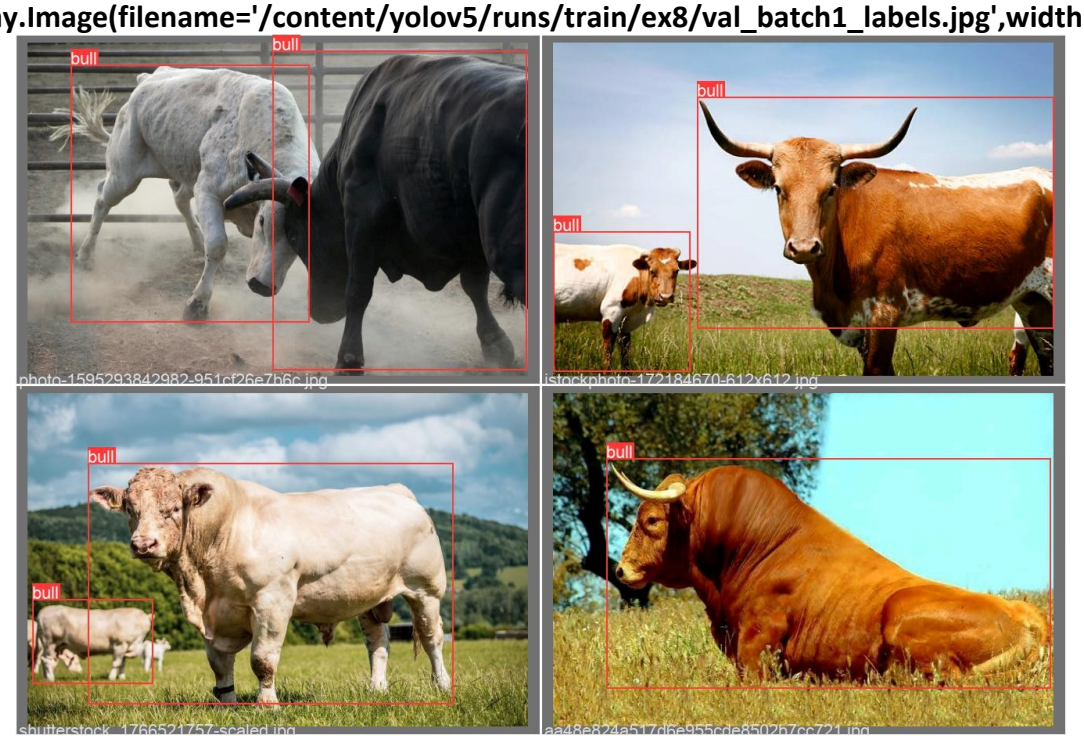
Training Screenshot is as follows:



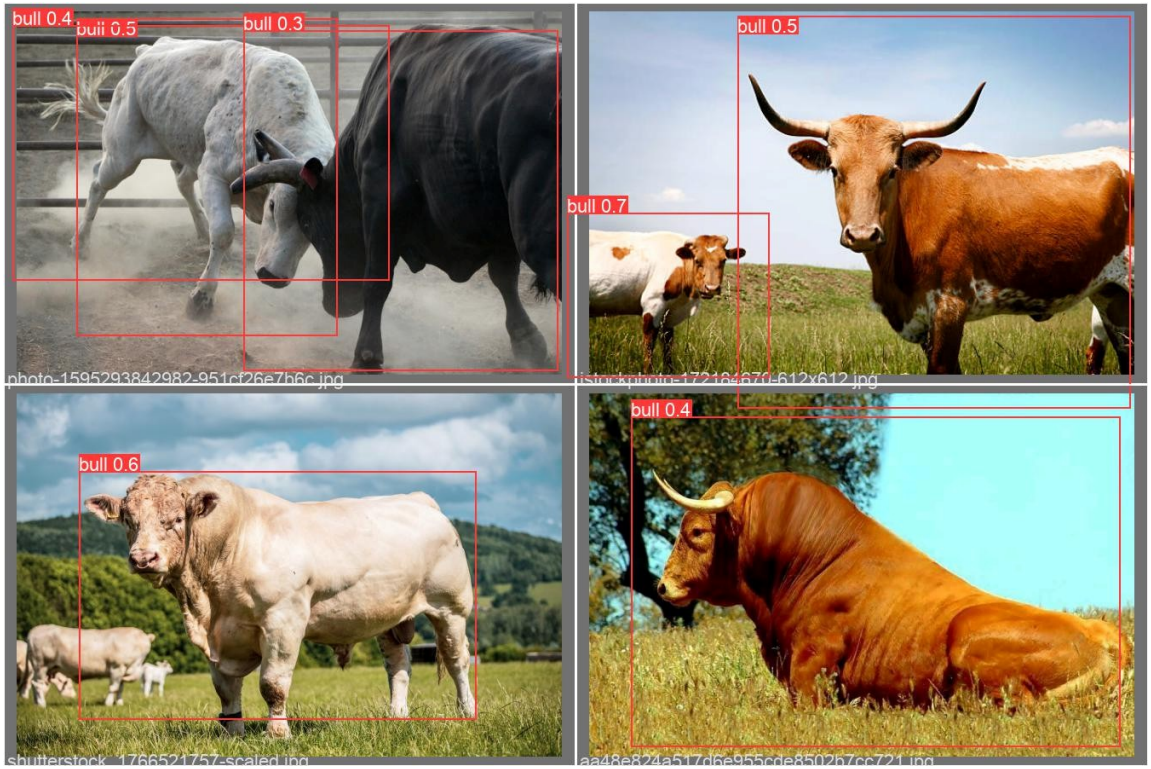
STEP 13: RESULTS

All results are logged by default to runs/train, with a new experiment directory created for each new training as runs/train/exp19, runs/train/exp8, etc. View train and val jpgs to see labels, and effects.

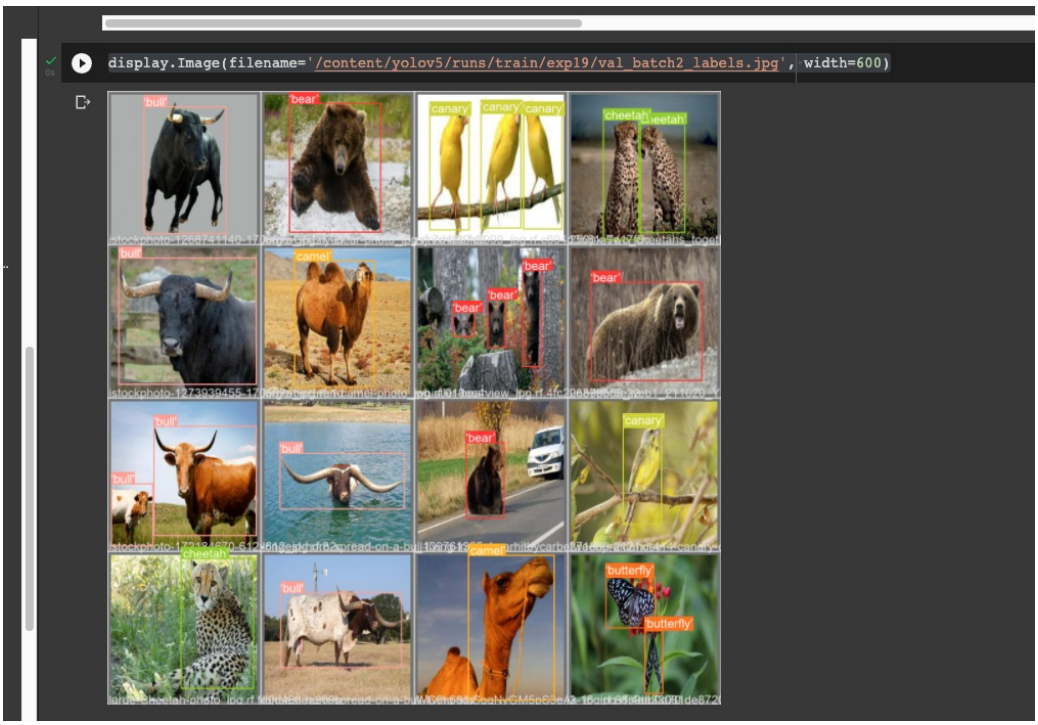
display.Image(filename='/content/yolov5/runs/train/ex8/val_batch1_labels.jpg',width=600)



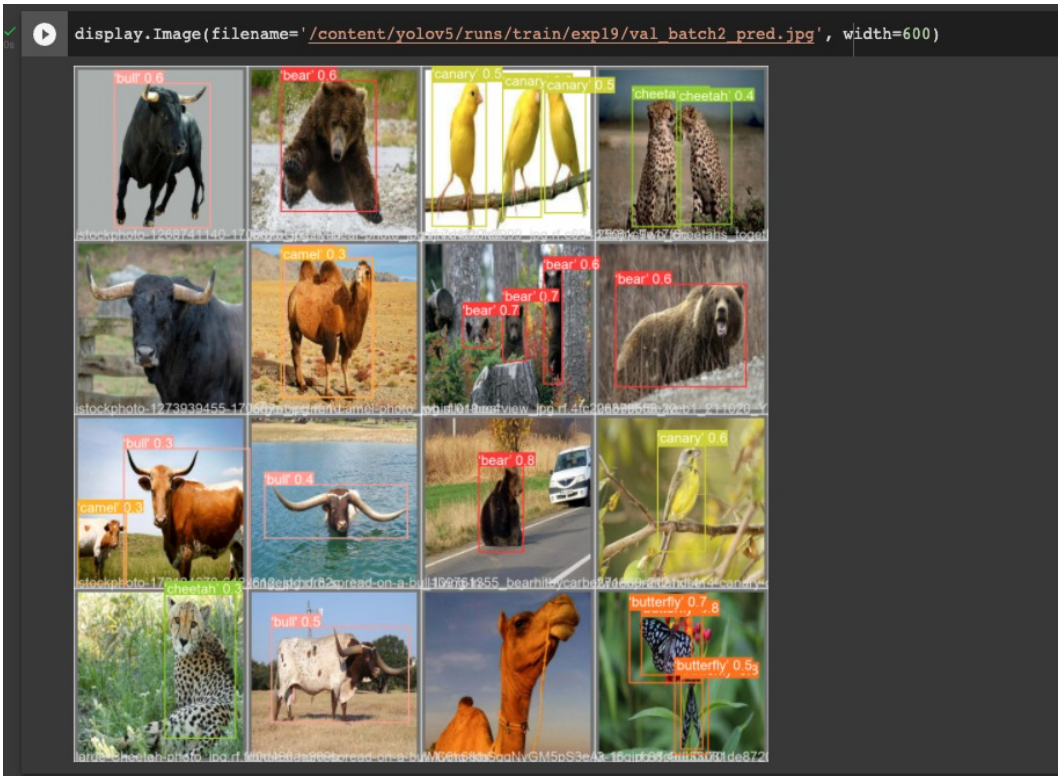
display.Image(filename='/content/yolov5/runs/train/exp8/val_batch1_pred.jpg', width=600)



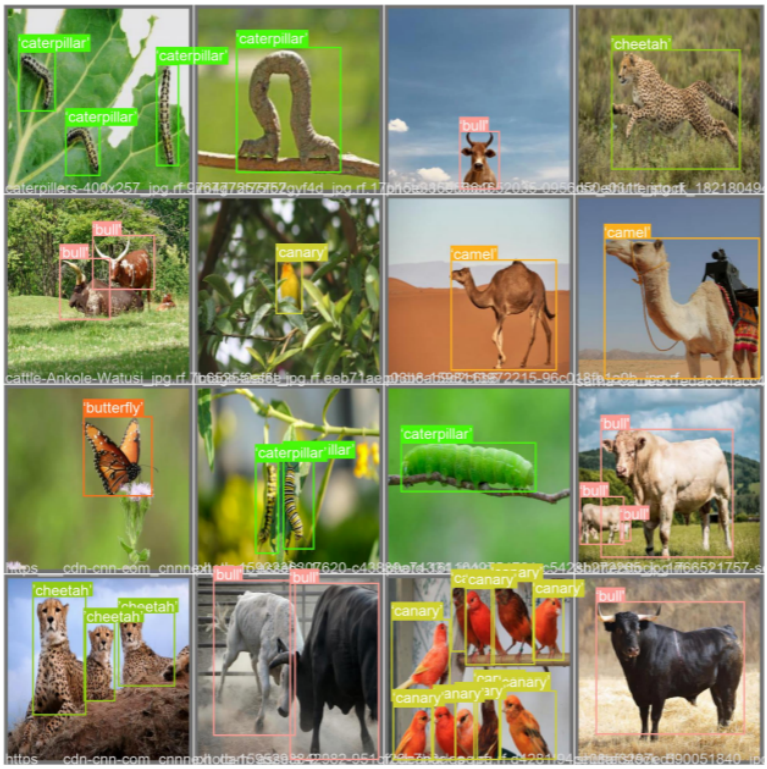
display.Image(filename='/content/yolov5/runs/train/exp19/val_batch2_labels.jpg',width=600)



display.Image(filename='/content/yolov5/runs/train/exp19/val_batch2_pred.jpg', width=600)



display.Image(filename='/content/yolov5/runs/train/exp19/val_batch1_labels.jpg',width=600)

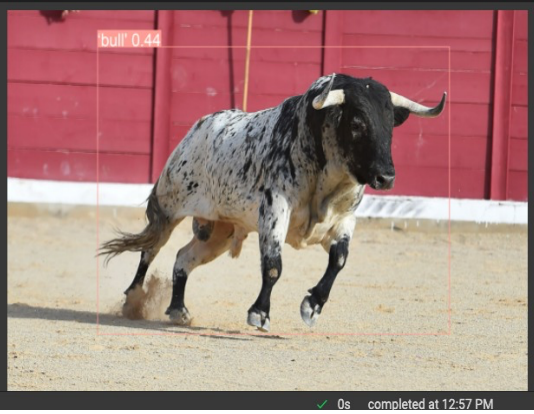


display.Image(filename='/content/yolov5/runs/train/exp19/val_batch0_pred.jpg', width=600)

<u>DETECTION OF CUSTOM ADDED IMAGE:</u>

!python detect.py –weights best.pt --img 640 – conf 0.25 –source /content/GettyImages-1131028486-a553aa259a09469a9b9469…jpg

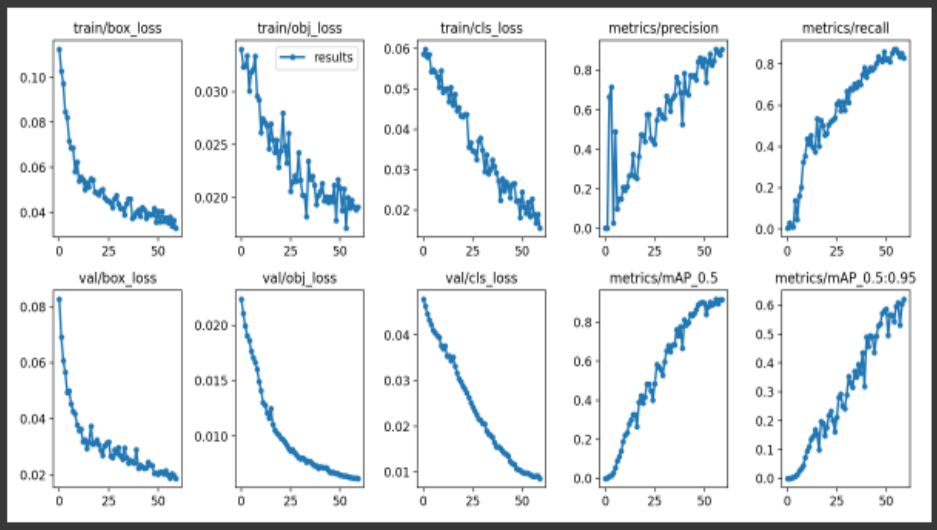
```
[56] !python detect.py --weights best.pt --img 640 --conf 0.25 --source /content/GettyImages-1131028486-a553aa259e09469a9b9460b946526e64.jpg
     #display.Image(filename='/content/iStock_000007001958Small-e1359525582279.jpg', width=600)

     detect: weights=['best.pt'], source=/content/GettyImages-1131028486-a553aa259e09469a9b9460b946526e64.jpg, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_d
     YOLOv5 🚀 v6.0-148-gc1249a4 torch 1.10.0+cu111 CUDA:0 (Tesla K80, 11441MiB)

     Fusing layers...
     Model Summary: 213 layers, 7029004 parameters, 0 gradients, 15.9 GFLOPs
     image 1/1 /content/GettyImages-1131028486-a553aa259e09469a9b9460b946526e64.jpg: 448x640 1 'bull', Done. (0.030s)
     Speed: 0.6ms pre-process, 30.4ms inference, 1.7ms NMS per image at shape (1, 3, 640, 640)
     Results saved to runs/detect/exp9

display.Image(filename='/content/yolov5/runs/detect/exp8/GettyImages-1131028486-a553aa259e09469a9b9460b946526e64.jpg', width=600)
```



0s completed at 12:57 PM

<u>TRAINING RESULTS PLOTS:</u>

```
display.Image(filename='/content/yolov5/runs/train/exp2/results.png', width=600)
```



This shows that the most out of the given data has been trained to around 95% if we validate, also this can be improved by providing more data and by improving the labeling quality. Overall,

the project was very well trained and as u can see in the mAP, that most of the data lies near to 1. Quality of model can me increased by training for more epochs.

CONCLUSION

YOLOv5 can achieve fast detection at 140FPS running on Tesla with weight files which are very small compared to the older models. The data was successfully trained and animal detection was achieved using the trained model. Predicted and labeled images are mention in the results and the predicted detection can be seen. Some test images were uploaded and detected the output most times, the object probability varied from 0.4-0.9 and the model was able to detect the specified object i.e., animal. Test video was also uploaded to detect the animal in the video which was also very accurate.

WAY FORWARD

Way forward to this project can be seen as implementation of this model to local cctv cameras in order to know and track the wild animals in the residential areas. Other than that, defining vicinity of your pet to notify the owner if the pet is there in the room or not, while taking care of any abnormal activity. Can be deployed in farmlands to notify the farmers if their crops are in danger.

REFERENCES

1. Chen, X. (2008). Application of Matlab in Moving Object Detecting Algorithm. 2008 International Seminar on Future Biomedical Information Engineering.
2. B. Gupta, A. Chaube, A. Negi, & U. Goel. Study on Object Detection using OpenCV - Python. International Journal of Computer Applications (0975 – 8887) Volume 162 – No 8, March 2017.
3. Antoreepjana. (2021, May 5). *Animals Detection Images Dataset*. Kaggle. Retrieved December 17, 2021, from https://www.kaggle.com/antoreepjana/animals-detection-images-dataset/code
4. Solawetz, J. (2020, September 29). *How to train a custom object detection model with Yolo V5*. Medium. Retrieved December 17, 2021, from https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-yolo-v5-917e9ce13208
5. Solawetz, J. (2021, October 27). *How to train yolov5 on a custom dataset*. Roboflow Blog. Retrieved December 17, 2021, from https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/?ref=ultralytics
6. Agarwal, R. (2020, October 19). *How to create an end to end object detector using yolov5*. Medium. Retrieved December 17, 2021, from https://towardsdatascience.com/how-to-create-an-end-to-end-object-detector-using-yolov5-35fbb1a02810