Install Dependencies

(Remember to choose GPU in Runtime if not already selected. Runtime --> Change Runtime Type --> Hardware accelerator --> GPU)

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

```
Cloning into 'yolov5'...
remote: Enumerating objects: 9791, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 9791 (delta 5), reused 9 (delta 4), pack-reused 9778
Receiving objects: 100% (9791/9791), 9.99 MiB | 26.64 MiB/s, done.
Resolving deltas: 100% (6809/6809), done.
/content/yolov5
HEAD is now at 886f1c0 DDP after autoanchor reorder (#2421)
```

```
# 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_
```

```
     |████████████████████████████████| 636 kB 5.0 MB/s
Setup complete. Using torch 1.9.0+cu111 _CudaDeviceProperties(name='Tesla P10
```

```
#follow the link below to get your download code from from Roboflow
!pip install -q roboflow
from roboflow import Roboflow
rf = Roboflow(model_format="yolov5", notebook="roboflow-yolov5")
```

```
     |████████████████████████████████| 178 kB 6.3 MB/s
     |████████████████████████████████| 1.1 MB 34.4 MB/s
     |████████████████████████████████| 138 kB 63.5 MB/s
     |████████████████████████████████| 62 kB 1.1 MB/s
  Building wheel for roboflow (setup.py) ... done
  Building wheel for wget (setup.py) ... done
ERROR: pip's dependency resolver does not currently take into account all the
google-colab 1.0.0 requires requests~=2.23.0, but you have requests 2.26.0 wh
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is
albumentations 0.1.12 requires imgaug<0.2.7,>=0.2.5, but you have imgaug 0.2.
upload and label your dataset, and get an API KEY here: https://app.roboflow.
```

```
%cd /content/yolov5
#after following the link above, recieve python code with these fields filled i
#from roboflow import Roboflow
#rf = Roboflow(api_key="YOUR API KEY HERE")
#project = rf.workspace().project("YOUR PROJECT")
#dataset = project.version("YOUR VERSION").download("yolov5")
```

```
/content/yolov5
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in American-Mushrooms-1 to yolov5pytorch: 100
Extracting Dataset Version Zip to American-Mushrooms-1 in yolov5pytorch:: 100
```

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

```
names:
- CoW
- chanterelle
nc: 2
train: American-Mushrooms-1/train/images
val: American-Mushrooms-1/valid/images
```

Define Model Configuration and Architecture

We will write a yaml script that defines the parameters for our model like the number of classes, anchors, and each layer.

You do not need to edit these cells, but you may.

```
# 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'])
```

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

```
#customize iPython writefile so we can write variables
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()))
```

```
%%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# parameters
nc: {num_classes}  # 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, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [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, BottleneckCSP, [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, BottleneckCSP, [256, False]],  # 17 (P3/8-small)

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

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

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

Train Custom YOLOv5 Detector

Next, we'll fire off training!

Here, we are able to pass a number of arguments:

- **img:** define input image size
- **batch:** determine batch size
- **epochs:** define the number of training epochs. (Note: often, 3000+ are common here!)
- **data:** set the path to our yaml file
- **cfg:** specify our model configuration

- **weights:** specify a custom path to weights. (Note: you can download weights from the Ultralytics Google Drive folder)
- **name:** result names
- **nosave:** only save the final checkpoint
- **cache:** cache images for faster training

```
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 100 --data {dataset.location}/da
```

```
/content/yolov5
github: ⚠ WARNING: code is out of date by 577 commits. Use 'git pull' to up
YOLOv5 v4.0-126-g886f1c0 torch 1.9.0+cu111 CUDA:0 (Tesla P100-PCIE-16GB, 16

Namespace(adam=False, batch_size=16, bucket='', cache_images=True, cfg='./m
wandb: Install Weights & Biases for YOLOv5 logging with 'pip install wandb'
Start Tensorboard with "tensorboard --logdir runs/train", view at http://lo
hyperparameters: lr0=0.01, lrf=0.2, momentum=0.937, weight_decay=0.0005, wa

                 from  n    params  module
  0                -1  1      3520  models.common.Focus
  1                -1  1     18560  models.common.Conv
  2                -1  1     19904  models.common.BottleneckCSP
  3                -1  1     73984  models.common.Conv
  4                -1  1    161152  models.common.BottleneckCSP
  5                -1  1    295424  models.common.Conv
  6                -1  1    641792  models.common.BottleneckCSP
  7                -1  1   1180672  models.common.Conv
  8                -1  1    656896  models.common.SPP
  9                -1  1   1248768  models.common.BottleneckCSP
 10                -1  1    131584  models.common.Conv
 11                -1  1         0  torch.nn.modules.upsampling.Upsample
 12           [-1, 6]  1         0  models.common.Concat
 13                -1  1    378624  models.common.BottleneckCSP
 14                -1  1     33024  models.common.Conv
 15                -1  1         0  torch.nn.modules.upsampling.Upsample
 16           [-1, 4]  1         0  models.common.Concat
 17                -1  1     95104  models.common.BottleneckCSP
 18                -1  1    147712  models.common.Conv
 19          [-1, 14]  1         0  models.common.Concat
 20                -1  1    313088  models.common.BottleneckCSP
 21                -1  1    590336  models.common.Conv
 22          [-1, 10]  1         0  models.common.Concat
 23                -1  1   1248768  models.common.BottleneckCSP
 24      [17, 20, 23]  1     18879  models.yolo.Detect
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:718: UserWarn
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ce
Model Summary: 283 layers, 7257791 parameters, 7257791 gradients, 16.8 GFLO

Scaled weight_decay = 0.0005
Optimizer groups: 62 .bias, 70 conv.weight, 59 other
train: Scanning 'American-Mushrooms-1/train/labels' for images and labels..
train: New cache created: American-Mushrooms-1/train/labels.cache
train: Caching images (0.1GB): 100% 123/123 [00:00<00:00, 511.60it/s]
val: Scanning 'American-Mushrooms-1/valid/labels' for images and labels... 5
```

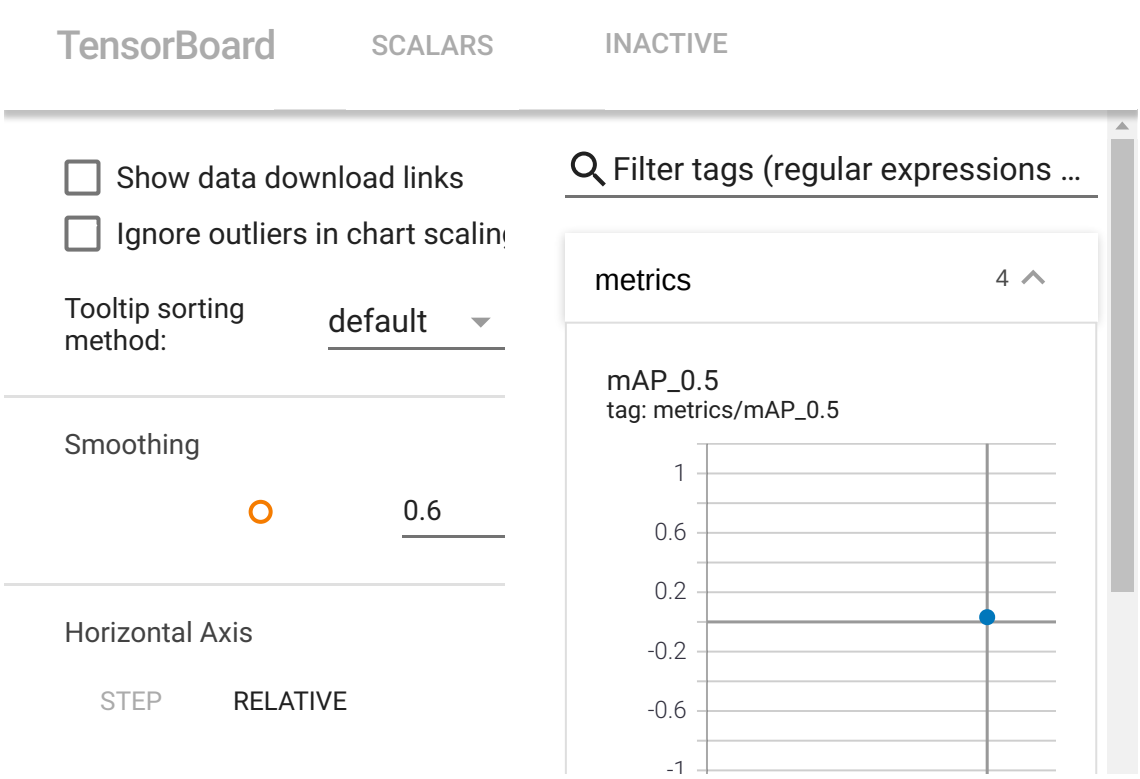
```
val: New cache created: American-Mushrooms-1/valid/labels.cache
val: Caching images (0.0GB): 100% 5/5 [00:00<00:00, 321.12it/s]
[W pthreadpool-cpp.cc:90] Warning: Leaking Caffe2 thread-pool after fork. (
[W pthreadpool-cpp.cc:90] Warning: Leaking Caffe2 thread-pool after fork. (
Plotting labels...

autoanchor: Analyzing anchors... anchors/target = 3.48, Best Possible Recal
Image sizes 416 train, 416 test
Using 2 dataloader workers
Logging results to runs/train/yolov5s_results
Starting training for 100 epochs...
```

Evaluate Custom YOLOv5 Detector Performance

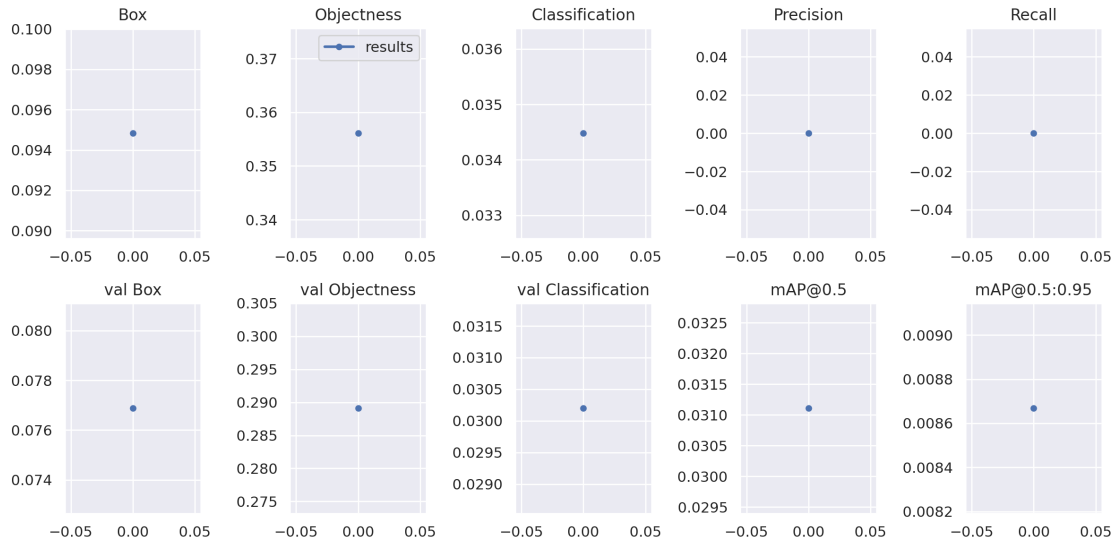
Training losses and performance metrics are saved to Tensorboard and also to a logfile defined above with the **--name** flag when we train. In our case, we named this `yolov5s_results`. (If given no name, it defaults to `results.txt`.) The results file is plotted as a png after training completes.

Note from Glenn: Partially completed `results.txt` files can be plotted with `from utils.utils import plot_results; plot_results()`.

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```




```
# we can also output some older school graphs if the tensor board isn't working
from utils.plots import plot_results  # plot results.txt as results.png
Image(filename='/content/yolov5/runs/train/yolov5s_results/results.png', width=
```



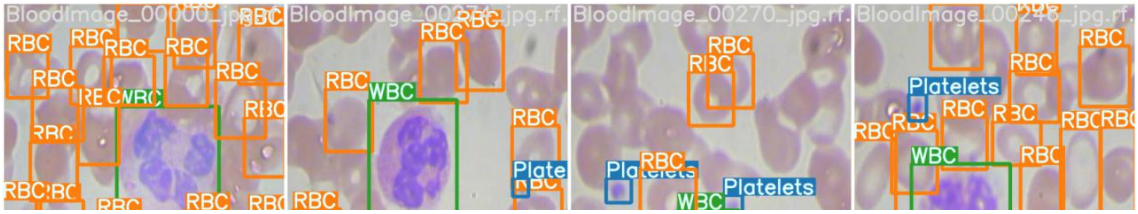
Curious? Visualize Our Training Data with Labels

After training starts, view `train*.jpg` images to see training images, labels and augmentation effects.

Note a mosaic dataloader is used for training (shown below), a new dataloading concept developed by Glenn Jocher and first featured in YOLOv4.

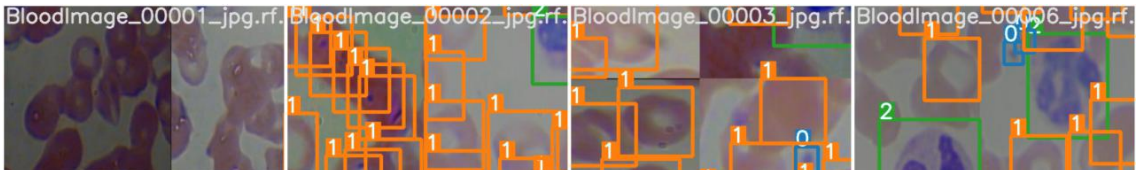
```
# first, display our ground truth data
print("GROUND TRUTH TRAINING DATA:")
Image(filename='/content/yolov5/runs/train/yolov5s_results/test_batch0_labels.j
```

GROUND TRUTH TRAINING DATA:




```
# print out an augmented training example
print("GROUND TRUTH AUGMENTED TRAINING DATA:")
Image(filename='/content/yolov5/runs/train/yolov5s_results/train_batch0.jpg', w
```

```
GROUND TRUTH AUGMENTED TRAINING DATA:
```



Run Inference With Trained Weights

Run inference with a pretrained checkpoint on contents of `test/images` folder downloaded from Roboflow.

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

```
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 TESl
# use the best weights!
%cd /content/yolov5/
!python detect.py --weights runs/train/yolov5s_results/weights/best.pt --img 41(
```

```
/content/yolov5
Namespace(agnostic_nms=False, augment=False, classes=None, conf_thres=0.4, de
Using torch 1.7.1+cu101 CPU

Fusing layers...
Model Summary: 232 layers, 7251912 parameters, 0 gradients
image 1/36 /content/test/images/BloodImage_00038_jpg.rf.ffa23e4b5b55b523367f3
image 2/36 /content/test/images/BloodImage_00044_jpg.rf.e7760375eba4bc20c5746
image 3/36 /content/test/images/BloodImage_00062_jpg.rf.1be1ca0ecdf783798fc10
image 4/36 /content/test/images/BloodImage_00090_jpg.rf.cdbf8f6ed3b93fa902a0b
image 5/36 /content/test/images/BloodImage_00099_jpg.rf.e3c42cd68359527494a53
image 6/36 /content/test/images/BloodImage_00112_jpg.rf.978cec39235980055c2ad
image 7/36 /content/test/images/BloodImage_00113_jpg.rf.a17463f1ddc2e7729f935
image 8/36 /content/test/images/BloodImage_00120_jpg.rf.6725d54bf5615683448eb
image 9/36 /content/test/images/BloodImage_00133_jpg.rf.39ee4e4a097a7b40defa5
image 10/36 /content/test/images/BloodImage_00134_jpg.rf.0d9da503b62e0034a281
image 11/36 /content/test/images/BloodImage_00154_jpg.rf.e5b45569e9cbede1ed36
image 12/36 /content/test/images/BloodImage_00160_jpg.rf.894e8c8c7179ec9958a4
image 13/36 /content/test/images/BloodImage_00190_jpg.rf.3e0bf272a5f8ea902775
image 14/36 /content/test/images/BloodImage_00191_jpg.rf.313648dd345edbdd0058
image 15/36 /content/test/images/BloodImage_00204_jpg.rf.04ba9998769a12d374e6
image 16/36 /content/test/images/BloodImage_00227_jpg.rf.d1790b0cdc042312d1e0
image 17/36 /content/test/images/BloodImage_00235_jpg.rf.283fff79a1188b82f5a5
image 18/36 /content/test/images/BloodImage_00241_jpg.rf.757020b43fe3414a0f66
image 19/36 /content/test/images/BloodImage_00254_jpg.rf.e95bc889425924cc7a35
image 20/36 /content/test/images/BloodImage_00265_jpg.rf.7d102d5f38caddb90a46
image 21/36 /content/test/images/BloodImage_00266_jpg.rf.2521623f1047a9502b27
image 22/36 /content/test/images/BloodImage_00275_jpg.rf.585f5abcf10f926c74ac
```

```
image 23/36 /content/test/images/BloodImage_00278_jpg.rf.cfe491c301184766df6a
image 24/36 /content/test/images/BloodImage_00284_jpg.rf.63402cee4454cb6d7655
image 25/36 /content/test/images/BloodImage_00289_jpg.rf.72f51d668e5ebc31700b
image 26/36 /content/test/images/BloodImage_00301_jpg.rf.885ee9fbea0573ba35b9
image 27/36 /content/test/images/BloodImage_00302_jpg.rf.911302bb5caaf9467e10
image 28/36 /content/test/images/BloodImage_00325_jpg.rf.c8ab9fc71ad718a95901
image 29/36 /content/test/images/BloodImage_00334_jpg.rf.8b2c2a9c5a39e78eb74a
image 30/36 /content/test/images/BloodImage_00336_jpg.rf.8d3e710e8696c10bcbb1
image 31/36 /content/test/images/BloodImage_00337_jpg.rf.b6cb228440b9158cafec
image 32/36 /content/test/images/BloodImage_00350_jpg.rf.e2c472c90de4ce51fe4f
image 33/36 /content/test/images/BloodImage_00359_jpg.rf.03331bc3903822adbe98
image 34/36 /content/test/images/BloodImage_00369_jpg.rf.c3818de705d661536ed7
image 35/36 /content/test/images/BloodImage_00385_jpg.rf.865551dd16b189945cbb
image 36/36 /content/test/images/BloodImage_00386_jpg.rf.c708422b2d9c642f200a
Results saved to runs/detect/exp2
Done. (6.319s)
```

```
#display inference on ALL test images
#this looks much better with longer training above

import glob
from IPython.display import Image, display

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

Export Trained Weights for Future Inference

Now that you have trained your custom detector, you can export the trained weights you have made here for inference on your device elsewhere

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

```
Mounted at /content/gdrive
```

```
%cp /content/yolov5/runs/train/yolov5s_results/weights/best.pt /content/gdrive/I
```

Congrats!

Hope you enjoyed this!

--Team Roboflow