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.git  # clone repo
%cd yolov5
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 16413, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 16413 (delta 0), reused 4 (delta 0), pack-reused 16408
Receiving objects: 100% (16413/16413), 14.96 MiB | 10.23 MiB/s, done.
Resolving deltas: 100% (11263/11263), done.
/content/yolov5
```

```
# 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() el
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.4/196.4 kB 5.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 709.5/709.5 kB 14.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 8.1 MB/s eta 0:00:00
Setup complete. Using torch 2.1.0+cu121 _CudaDeviceProperties(name='Tesla T4', major=7, minor=5, total_memory=15102MB, multi_pr
```

Your Custom Data

```
# Export code snippet and paste here
%cd /content
!wget https://github.com/aadityaadyot/CV-Hand-Signal-Project/raw/main/Data/Sign_language_data.zip
!unzip -q Sign_language_data.zip
```

```
/content
--2024-02-09 20:33:29--  https://github.com/aadityaadyot/CV-Hand-Signal-Project/raw/main/Data/Sign_language_data.zip
Resolving github.com (github.com)... 192.30.255.112
Connecting to github.com (github.com)|192.30.255.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/aadityaadyot/CV-Hand-Signal-Project/main/Data/Sign_language_data.zip [following]
--2024-02-09 20:33:29--  https://raw.githubusercontent.com/aadityaadyot/CV-Hand-Signal-Project/main/Data/Sign_language_data.zip
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 39777787 (38M) [application/zip]
Saving to: ‘Sign_language_data.zip’

Sign_language_data. 100%[===================>]  37.93M  --.-KB/s    in 0.1s

2024-02-09 20:33:30 (303 MB/s) - ‘Sign_language_data.zip’ saved [39777787/39777787]
```

```
# this is the YAML file  we're loading into this notebook with our data
%cat data.yaml
```

```
train: ../train/images
val: ../test/images

nc: 7
names: ['Fuck Off', 'Hello', 'ILoveYou', 'Namaste', 'Please', 'Thanks', 'Yes']
```

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("data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])


num_classes
```

```
'7'
```

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

```
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license

# Parameters
nc: 80 # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.50 # layer channel multiple
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 v6.0 backbone
backbone:
  # [from, number, module, args]
  [
    [-1, 1, Conv, [64, 6, 2, 2]], # 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, 6, 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, 3, C3, [1024]],
        [-1, 1, SPPF, [1024, 5]], # 9
      ]

    # YOLOv5 v6.0 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 545 --batch 16 --epochs 500 --data '../data.yaml' --cfg ./models/custom_yolov5s.yaml --
```

```
                 all         37         37       0.97          1      0.977      0.832

500 epochs completed in 0.438 hours.
Optimizer stripped from runs/train/yolov5s_results/weights/last.pt, 14.8MB
Optimizer stripped from runs/train/yolov5s_results/weights/best.pt, 14.8MB

Validating runs/train/yolov5s_results/weights/best.pt...
Fusing layers...
custom_YOLOv5s summary: 182 layers, 7262700 parameters, 0 gradients
                Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2 [00:00<00:00,  3.25it/s]
                  all         37         37      0.968          1      0.977       0.85
             Fuck Off         37          4      0.984          1      0.995      0.937
                Hello         37          5      0.984          1      0.995      0.879
             ILoveYou         37          7      0.865          1      0.871      0.732
              Namaste         37          6      0.993          1      0.995      0.736
               Please         37          4      0.985          1      0.995       0.92
               Thanks         37          6      0.986          1      0.995      0.885
                  Yes         37          5      0.982          1      0.995      0.862
Results saved to runs/train/yolov5s_results
CPU times: user 16.8 s, sys: 1.98 s, total: 18.8 s
Wall time: 26min 52s
```

Evaluate Custom YOLOv5 Detector Performance

Double-click (or enter) to edit

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
```

TensorBoard TIME SERIES SCALARS IMAGES INACTIVE

Filter runs (regex)

Run ↑

train/yolov5s_results

Filter tags (regex) All Scalars Image Histogram Settings

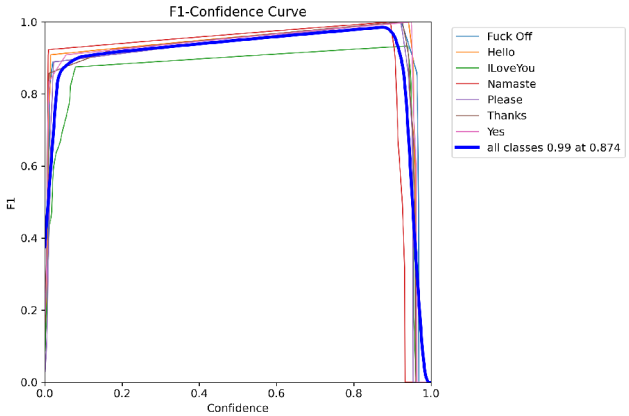
Pinned

Pin cards for a quick view and comparison

F1_curve

F1_curve

train/yolov5s_resul... Step 499



PR_curve

PR_curve

train/yolov5s_resul... Step 499

Settings

GENERAL

Horizontal Axis

Step

Enable step selection and data table

(Scalars only)

Enable Range Selection

Link by step 499

Card Width

SCALARS

Smoothing

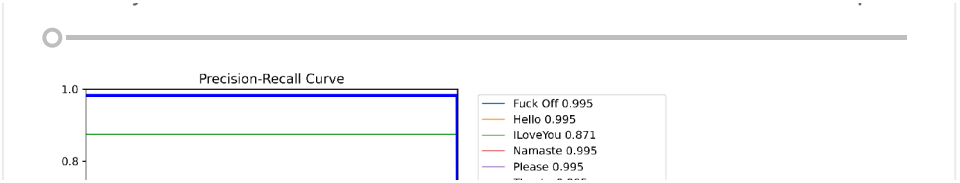
0.6

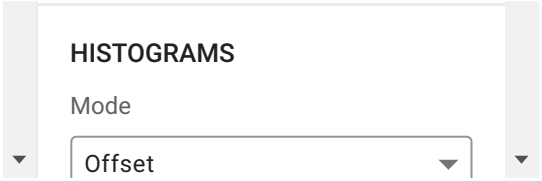
Tooltip sorting method

Alphabetical

Ignore outliers in chart scaling

Partition non-monotonic X axis



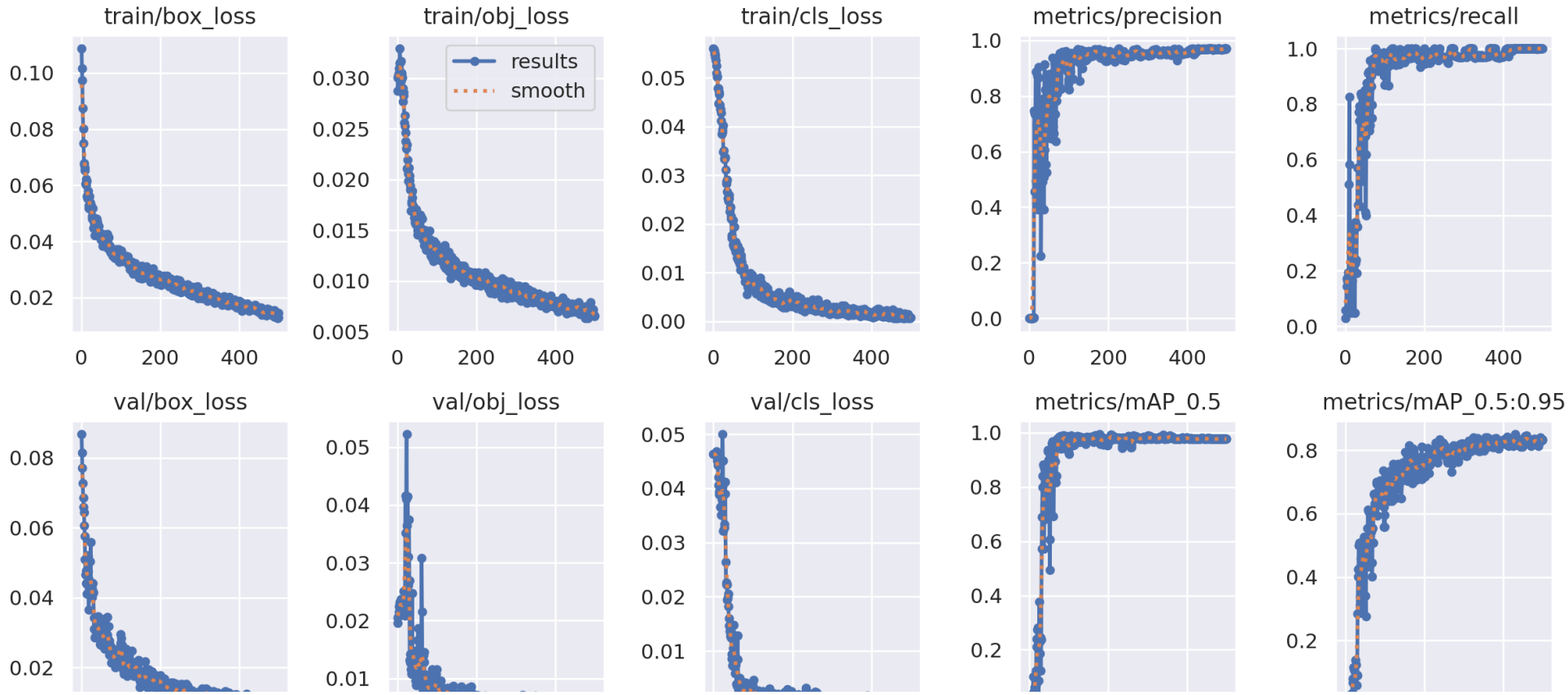



```
!nvidia-smi
```

```
Fri Feb  9 21:11:57 2024
+---------------------------------------------------------------------------------------+
| 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   49C    P8              10W /  70W |      3MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

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

```
# we can also output some older school graphs if the tensor board isn't working for whatever reason...
from utils.plots import plot_results  # plot results.txt as results.png
Image(filename='/content/yolov5/runs/train/yolov5s_results/results.png', width=1000)  # view results.png
```



Curious? Visualize Our Training Data with Labels

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

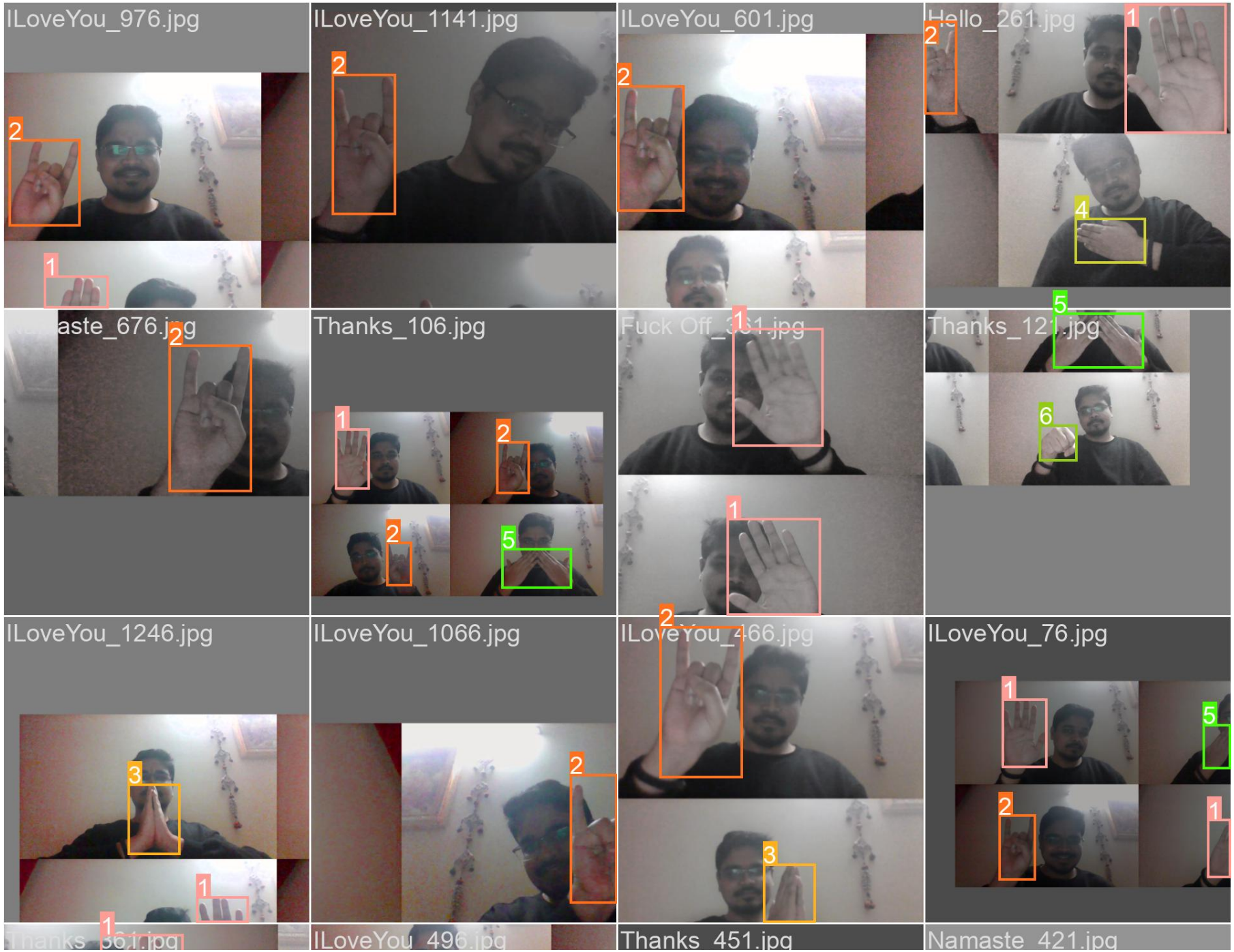
Note a mosaic dataloader is used for training (shown below), a new dataloading concept developed by Glenn Jocher and first featured in YOLOv4.

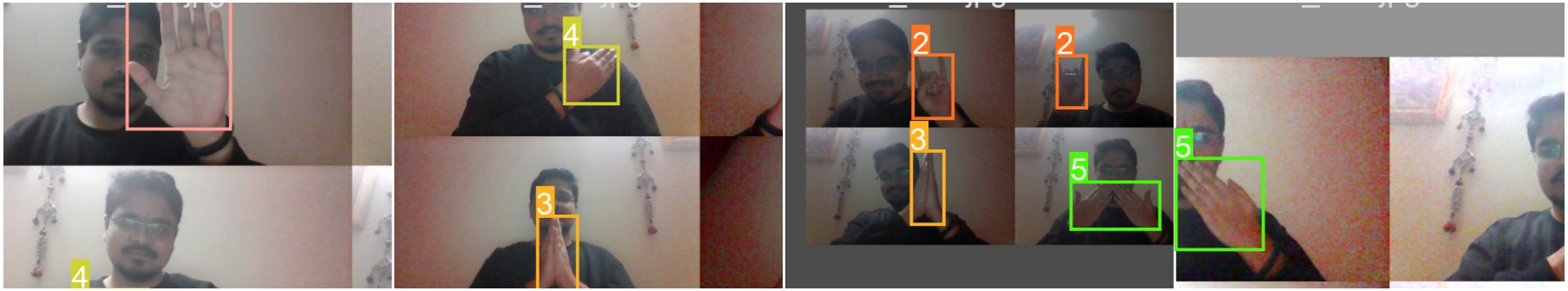
```
# first, display our ground truth data
print("GROUND TRUTH TRAINING DATA:")
Image(filename='/content/yolov5/runs/train/yolov5s_results/val_batch0_labels.jpg', width=900)
```

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', width=900)
```

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
```

Double-click (or enter) to edit

```
# 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 545 --conf 0.5 --source ../test/
```

```
/content/yolov5
detect: weights=['runs/train/yolov5s_results/weights/best.pt'], source=../test/images, data=data/coco128.yaml, imgsz=[545, 545]
YOLOv5 🚀 v7.0-284-g95ebf68f Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
```

```
Fusing layers...
custom_YOLOv5s summary: 182 layers, 7262700 parameters, 0 gradients
WARNING ⚠ --img-size [545, 545] must be multiple of max stride 32, updating to [576, 576]
WARNING ⚠ NMS time limit 0.550s exceeded
image 1/37 /content/test/images/Fuck Off_1.jpg: 352x576 1 Fuck Off, 47.7ms
image 2/37 /content/test/images/Fuck Off_16.jpg: 352x576 1 Fuck Off, 9.0ms
image 3/37 /content/test/images/Fuck Off_31.jpg: 352x576 1 Fuck Off, 7.9ms
image 4/37 /content/test/images/Fuck Off_46.jpg: 352x576 1 Fuck Off, 7.9ms
image 5/37 /content/test/images/Hello_121.jpg: 352x576 1 Hello, 7.9ms
image 6/37 /content/test/images/Hello_161.jpg: 352x576 1 Hello, 7.8ms
image 7/37 /content/test/images/Hello_21.jpg: 352x576 1 Hello, 7.8ms
image 8/37 /content/test/images/Hello_31.jpg: 352x576 1 Hello, 7.9ms
image 9/37 /content/test/images/Hello_61.jpg: 352x576 1 Hello, 8.8ms
image 10/37 /content/test/images/ILoveYou_136.jpg: 352x576 1 ILoveYou, 7.9ms
image 11/37 /content/test/images/ILoveYou_16.jpg: 352x576 1 ILoveYou, 7.9ms
```