Concurso Kaggle

Yolov5 (paso a paso)

1-Yolo v5 - https://www.kaggle.com/ultralytics/yolov5

YOLO (You Only Look Once) is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

2-Versiones Yolo importantes y links documentación

YOLOv5
Shortly after the release of YOLOv4 Glenn Jocher introduced YOLOv5 using the Pytorch framework.
The open source code is available on GitHub
Author: Glenn Jocher
Released: 18 May 2020 https://github.com/ultralytics/yolov5

YOLOv4
With the original authors work on YOLO coming to a standstill, YOLOv4 was released by Alexey Bochoknovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. The paper was titled YOLOv4: Optimal Speed and Accuracy of Object Detection
Author: Alexey Bochoknovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao
Released: 23 April 2020 Yolov4 https://arxiv.org/abs/2004.10934v1

3- https://www.kaggle.com/ultralytics/yolov5

Version 14 of 14

4 - Yolov5 en Kaggle 1: Setup

Clone repo, install dependencies and check PyTorch and GPU.

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

import torch
from IPython.display import Image, clear_output  # to display images

clear_output()
print(f"Setup complete. Using torch {torch.__version__}
({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

5 - yolov5 - 2. Inference

detect.py runs YOLOv5 inference on a variety of sources, downloading models automatically from the latest YOLOv5 release, and saving results to runs/detect

. Example inference sources are:

```
$ python detect.py --source 0  # webcam
                           file.jpg  # image
                           file.mp4  # video
                           path/  # directory
                           path/*.jpg  # glob
                           'https://youtu.be/NUsoVlDFqZg'  # YouTube video
                           'rtsp://example.com/media.mp4'  # RTSP, RTMP, HTTP stream
```

!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images/
Image(filename='runs/detect/exp/zidane.jpg', width=600)

5 - yolov5 - 2. Inference

```
detect: weights=['yolov5s.pt'], source=data/images/, imgsz=640, conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False,
save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False,
line_thickness=3, hide_labels=False, hide_conf=False, half=False
Downloading https://github.com/ultralytics/yolov5/releases/download/v5.0/yolov5s.pt to yolov5s.pt...
100%|| 14.1M/14.1M [00:00<00:00, 100MB/s]

image 1/2 /kaggle/working/yolov5/data/images/bus.jpg: 640x480 4 persons, 1 bus, 1 fire hydrant, Done. (0.043s)
image 2/2 /kaggle/working/yolov5/data/images/zidane.jpg: 384x640 2 persons, 2 ties, Done. (0.028s)
Results saved to runs/detect/exp
Done. (0.199s)
```

6- yolov5 - 3. Validate

Validate a model's accuracy on COCO val or test-dev datasets. Models are downloaded automatically from the latest YOLOv5 release. To show results by class use the --verbose flag. Note that pycocotools metrics may be ~1% better than the equivalent repo metrics, as is visible below, due to slight differences in mAP computation.

COCO val2017
Download COCO val 2017 dataset (1GB - 5000 images), and test model accuracy.

```
# Download COCO val2017
torch.hub.download_url_to_file('https://github.com/ultralytics/yolov5/releases/download/v1.0/coco2017val.zip', 'tmp.zip')
!unzip -q tmp.zip -d ../datasets && rm tmp.zip
# Run YOLOv5x on COCO val2017
!python val.py --weights yolov5x.pt --data coco.yaml --img 640 --iou 0.65 --half
```

6- yolov5 - 3. Validate

```
val: data=./data/coco.yaml, weights=['yolov5x.pt'], batch_size=32, imgsz=640, conf_thres=0.001,
iou_thres=0.65, task=val, device=, single_cls=False, augment=False, verbose=False, save_txt=False,
save_hybrid=False, save_conf=False, save_json=True, project=runs/val, name=exp, exist_ok=False,
half=True
Downloading https://github.com/ultralytics/yolov5/releases/download/v5.0/yolov5x.pt to yolov5x.pt...
100%|                                                                                | 168M/168M
[00:02<00:00, 81.6MB/s]

val: Scanning '../datasets/coco/val2017' images and labels...4952 found, 48 miss
               Class     Images     Labels          P          R     mAP@.5 mAP@val.py:61: UserWarning: This overload
of nonzero is deprecated:
                  nonzero()
Consider using one of the following signatures instead:
                  nonzero(*, bool as_tuple) (Triggered internally at  /opt/conda/conda-
bld/pytorch_1603729047590/work/torch/csrc/utils/python_arg_parser.cpp:882.)
  ti = (cls == tcls).nonzero().view(-1)  # label indices
               Class     Images     Labels          P          R     mAP@.5 mAP@
```

7- yolov5 - 4. COCO test-dev2017

Download COCO test2017 dataset (7GB - 40,000 images), to test model accuracy on test-dev set (20,000 images, no labels). Results are saved to a *.json file which should be zipped and submitted to the evaluation server at https://competitions.codalab.org/competitions/20794.

```
# Download COCO test-dev2017
torch.hub.download_url_to_file('https://github.com/ultralytics/yolov5/releases/download/v1.0/coco2017labels.zip', 'tmp.zip')
!unzip -q tmp.zip -d ../ && rm tmp.zip # unzip labels
!f="test2017.zip" && curl http://images.cocodataset.org/zips/$f -o $f && unzip -q $f && rm $f  # 7GB,  41k images
%mv ./test2017 ../coco/images  # move to /coco
# Run YOLOv5s on COCO test-dev2017 using --task test
!python val.py --weights yolov5s.pt --data coco.yaml --task test
```

https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

7- yolov5 - 4. COCO test-dev2017

https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

Train On Custom Data
1. Create dataset.yaml
COCO128 is an example small tutorial dataset composed of the first 128 images in COCO train2017. These same 128 images are used for both training and validation to verify our training pipeline is capable of overfitting. data/coco128.yaml, shown below, is the dataset config file that defines 1) the dataset root directory path and relative paths to train / val / test image directories (or *.txt files with image paths), 2) the number of classes nc and 3) a list of class names:

```
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
path: ../datasets/coco128  # dataset root dir
train: images/train2017  # train images (relative to 'path') 128 images
val: images/train2017  # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
nc: 80  # number of classes
names: [ 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',
        'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
        'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
        'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',
        'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
        'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',
        'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
        'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',
        'hair drier', 'toothbrush' ]  # class names
```

8- yolov5 - 5. Train

Download COCO128, a small 128-image tutorial dataset, start tensorboard and train YOLOv5s from a pretrained checkpoint for 3 epochs (note actual training is typically much longer, around 300-1000 epochs, depending on your dataset).

```
# Download COCO128
torch.hub.download_url_to_file('https://github.com/ultralytics/yolov5/releases/download/v1.0/coco128.zip', 'tmp.zip')
!unzip -q tmp.zip -d ../ && rm tmp.zip
```

9- yolov5 - 5. Train

Train a YOLOv5s model on COCO128 with --data coco128.yaml, starting from pretrained --weights yolov5s.pt, or from randomly initialized --weights '' --cfg yolov5s.yaml. Models are downloaded automatically from the latest YOLOv5 release, and COCO, COCO128, and VOC datasets are downloaded automatically on first use.

All training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

10- yolov5 - 5. Train

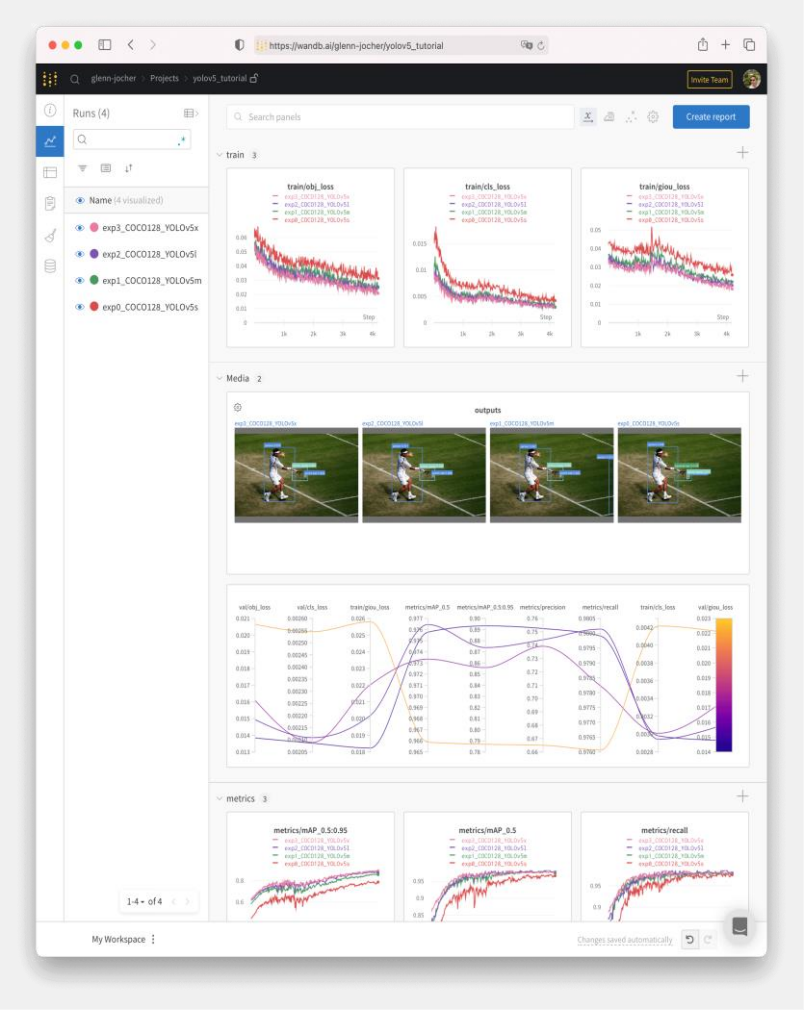
```
# Tensorboard  (optional)
%load_ext tensorboard
%tensorboard --logdir runs/train
# Weights & Biases  (optional)
%pip install -q wandb
import wandb
wandb.login()
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 3 --data coco128.yaml --weights yolov5s.pt --cache
```

11- yolov5 - Weights & Biases Logging

Weights & Biases (W&B) is now integrated with YOLOv5 for real-time visualization and cloud logging of training runs. This allows for better run comparison and introspection, as well improved visibility and collaboration for teams. To enable W&B pip install wandb, and then train normally (you will be guided through setup on first use).

During training you will see live updates at https://wandb.ai/home, and you can create and share detailed Reports of your results. For more information see the YOLOv5 Weights & Biases Tutorial.

12- yolov5 - Weights & Biases Logging

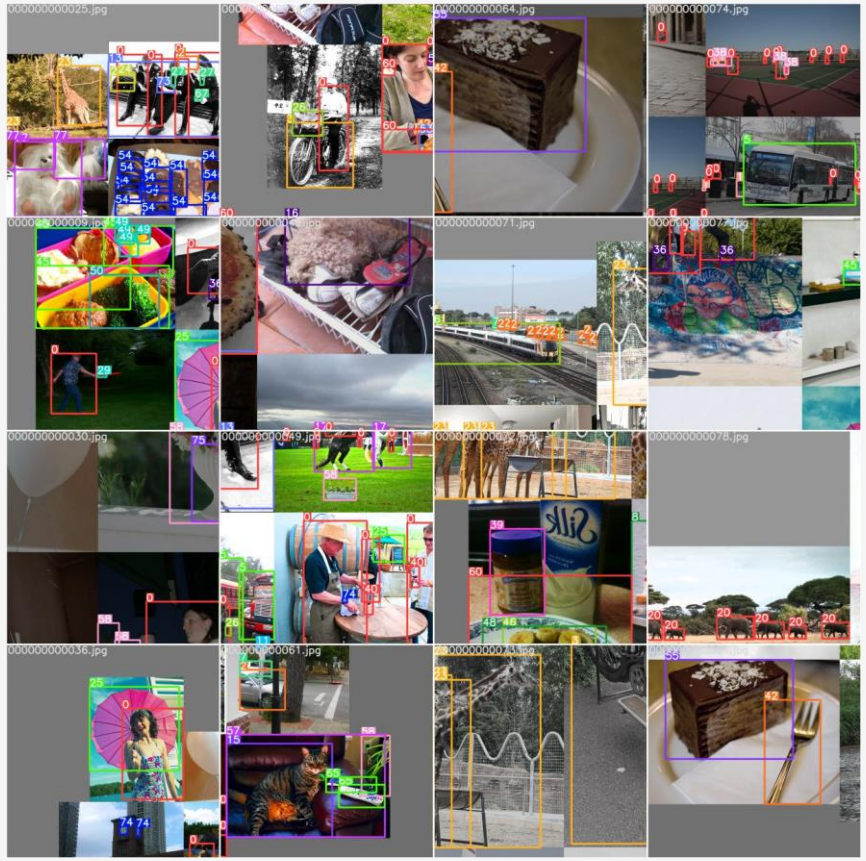


13- yolov5 - Local Logging

Local Logging
All results are logged by default to runs/train, with a new experiment directory created for each new training as runs/train/exp2, runs/train/exp3, etc. View train and val jpgs to see mosaics, labels, predictions and augmentation effects. Note a Mosaic Dataloader is used for training (shown below), a new concept developed by Ultralytics and first featured in YOLOv4.

```
Image(filename='runs/train/exp/train_batch0.jpg', width=800)  # train batch 0 mosaics and labels
Image(filename='runs/train/exp/test_batch0_labels.jpg', width=800)  # val batch 0 labels
Image(filename='runs/train/exp/test_batch0_pred.jpg', width=800)  # val batch 0 predictions
```

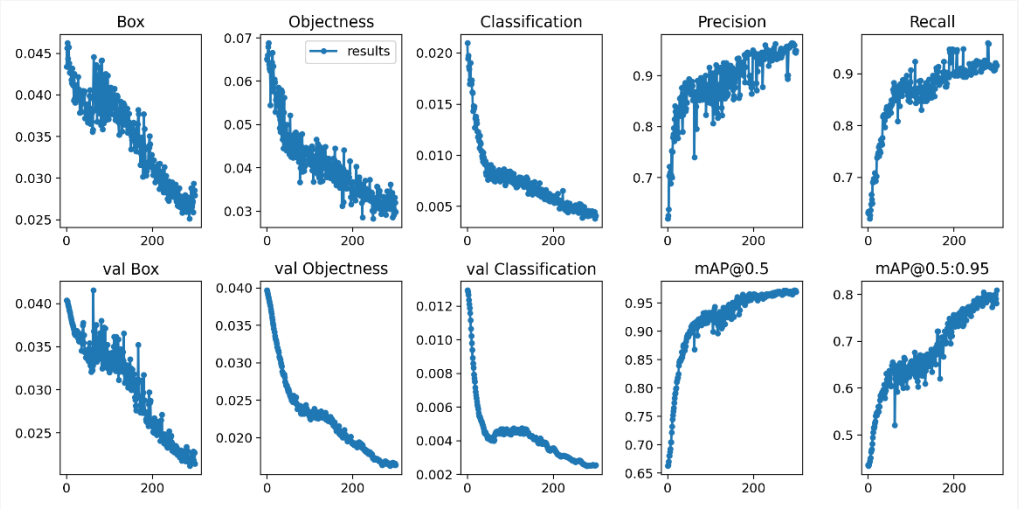
14- yolov5 - Mosaic Dataloader



Training results are automatically logged to Tensorboard and runs/train/exp/results.txt, which is plotted as results.png (below) after training completes.

15- yolov5 - You can also plot any results.txt file manually:

```
from utils.plots import plot_results
plot_results(save_dir='runs/train/exp')  # plot all results*.txt files in 'runs/train/exp'
Image(filename='runs/train/exp/results.png', width=800)
```



16- status

CI tests verify correct operation of YOLOv5 training (train.py), testing (val.py), inference (detect.py) and export (export.py) on MacOS, Windows, and Ubuntu every 24 hours and on every commit.

17- yolov5

Appendix¶
Optional extras below. Unit tests validate repo functionality and should be run on any PRs submitted.

```
# Reproduce
for x in 'yolov5s', 'yolov5m', 'yolov5l', 'yolov5x':
  !python val.py --weights {x}.pt --data coco.yaml --img 640 --conf 0.25 --iou 0.45  # speed
  !python val.py --weights {x}.pt --data coco.yaml --img 640 --conf 0.001 --iou 0.65  # mAP

# PyTorch Hub
import torch
# Model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
# Images
dir = 'https://ultralytics.com/images/'
imgs = [dir + f for f in ('zidane.jpg', 'bus.jpg')]  # batch of images
# Inference
results = model(imgs)
results.print()  # or .show(), .save()
```

18- yolov5

```
 Unit tests
%%shell
export PYTHONPATH="$PWD"  # to run *.py. files in subdirectories

rm -rf runs  # remove runs/
for m in yolov5s; do  # models
  python train.py --weights $m.pt --epochs 3 --img 320 --device 0  # train pretrained
  python train.py --weights '' --cfg $m.yaml --epochs 3 --img 320 --device 0  # train scratch
  for d in 0 cpu; do  # devices
    python detect.py --weights $m.pt --device $d  # detect official
    python detect.py --weights runs/train/exp/weights/best.pt --device $d  # detect custom
    python val.py --weights $m.pt --device $d # val official
    python val.py --weights runs/train/exp/weights/best.pt --device $d # val custom
  done
  python hubconf.py  # hub
  python models/yolo.py --cfg $m.yaml  # inspect
  python export.py --weights $m.pt --img 640 --batch 1  # export
done
```

19- yolov5

```
# Profile
from utils.torch_utils import profile

m1 = lambda x: x * torch.sigmoid(x)
m2 = torch.nn.SiLU()
profile(x=torch.randn(16, 3, 640, 640), ops=[m1, m2], n=100)
# Evolve
!python train.py --img 640 --batch 64 --epochs 100 --data coco128.yaml --weights
yolov5s.pt --cache --noautoanchor --evolve
!d=runs/train/evolve && cp evolve.* $d && zip -r evolve.zip $d && gsutil mv evolve.zip
gs://bucket  # upload results (optional)
# VOC
for b, m in zip([64, 48, 32, 16], ['yolov5s', 'yolov5m', 'yolov5l', 'yolov5x']):  #
zip(batch_size, model)
  !python train.py --batch {b} --weights {m}.pt --data VOC.yaml --epochs 50 --cache --img
512 --nosave --hyp hyp.finetune.yaml --project VOC --name {m}
```

20 # Introduction

This directory contains software developed by Ultralytics LLC, and **is freely available for redistribution under the GPL-3.0 license**. For more information please visit https://www.ultralytics.com.

Description

The https://github.com/ultralytics/COCO2YOLO repo contains code to convert JSON datasets into YOLO (darknet) format. The code works on Linux, MacOS and Windows.
Requirements
Python 3.7 or later with the following `pip3 install -U -r requirements.txt` packages:
- `numpy`
- `tqdm`

Citation
[![DOI](https://zenodo.org/badge/186122711.svg)](https://zenodo.org/badge/latestdoi/186122711)

Contact
Issues should be raised directly in the repository. For additional questions or comments please email Glenn Jocher at glenn.jocher@ultralytics.com or visit us at https://contact.ultralytics.com.

Concurso Kaggle

ejercicio (paso a paso)

https://www.kaggle.com/claudiorparrinello/siim-covid-19-yolo-v5-image-level-predictions/

1- paso1

1

```
#creating new directory
%cd ../
!mkdir siim_covid19
# Reason for creating & moving into this directory can be seen in above folder structure
%cd siim_covid19

!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
!pip install -r requirements.txt
# moving back to working directory
%cd ../
```

2- Salvando logs en W&B paso2

2

```
##-----------------
#Not saving logs into W&B
##-----------------
# Access WANDB account
#!pip install -q --upgrade wandb
#import wandb
#wandb.login()
```

3- Copiando yolov5 e instalando (# pip install -r requirements.txt)

```
# pip install -r requirements.txt

# base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.2
Pillow
PyYAML>=5.3.1
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
tqdm>=4.41.0

# logging -------------------------------------
tensorboard>=2.4.1
# wandb

# plotting ------------------------------------
seaborn>=0.11.0
pandas

# export --------------------------------------
# coremltools>=4.1
# onnx>=1.9.0
# scikit-learn==0.19.2  # for coreml quantization

# extras --------------------------------------
# Cython  # for pycocotools https://github.com/cocodataset/cocoapi/issues/172
# pycocotools>=2.0  # COCO mAP
# albumentations>=1.0.3
thop  # FLOPs computation
```

4- Copiando modelos yolov5

4

Inference with YOLOv5 and PyTorch Hub. Models automatically download from the latest YOLOv5 release.
https://github.com/ultralytics/yolov5/releases

```
# Model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')  # or yolov5m, yolov5l, yolov5x, custom

# Images
img = 'https://ultralytics.com/images/zidane.jpg'  # or PosixPath, PIL, OpenCV, numpy, list

# Inference
results = model(img)

# Results
results.print()  # or .show(), .save(), .crop(), .pandas(), etc.
```

5- Copiando yolov5 - Pretrained Checkpoints

Model	size (pixel)	mAP^{val} 0.5:0.95	mAP^{test} 0.5:0.95	mAP^{val} 0.5	Speed V100 (ms)	params (M)	FLOPS 640 (B)
YOLOv5s	640	36.7	36.7	55.4	**2.0**	7.3	17.0
YOLOv5m	640	44.5	44.5	63.1	2.7	21.4	51.3
YOLOv5l	640	48.2	48.2	66.9	3.8	47.0	115.4
YOLOv5x	640	**50.4**	**50.4**	**68.8**	6.1	87.7	218.8
YOLOv5s6	1280	43.3	43.3	61.9	**4.3**	12.7	17.4
YOLOv5m6	1280	50.5	50.5	68.7	8.4	35.9	52.4
YOLOv5l6	1280	53.4	53.4	71.1	12.3	77.2	117.7
YOLOv5x6	1280	**54.4**	**54.4**	**72.0**	22.4	141.8	222.9
YOLOv5x6 TTA	1280	**55.0**	**55.0**	**72.0**	70.8	-	

6- Inference with detect.py

6

detect.py runs inference on a variety of sources, downloading models automatically from the latest YOLOv5 release and saving results to runs/detect.

```
$ python detect.py --source 0  # webcam
                           file.jpg  # image
                           file.mp4  # video
                           path/  # directory
                           path/*.jpg  # glob
                           'https://youtu.be/NUsoVlDFqZg'  # YouTube video
                           'rtsp://example.com/media.mp4'  # RTSP, RTMP, HTTP stream
```

Importing Dependencies

```
import os
import shutil
import tensorflow as tf
import yaml

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import ast

import matplotlib.pyplot as plt
import cv2
import PIL

import warnings
warnings.filterwarnings('ignore')
```

Importing Dependencies

```
TRAIN_PATH = "../input/siim-covid19-dataset-256px-
jpg/512px/train/train/"
HEIGHT,WIDTH = 512,512
CHANNELS = 3
BATCH_SIZE = 16
EPOCHS  = 40
SEED  =2021
```

CSV file

```
# Get image path from image_id
def get_path(image_id):
    path = tf.io.gfile.glob(TRAIN_PATH + f"*{image_id}.jpg")[0]
    return path

image_dict = {
    "opacity" : 1,
    "none" : 0
}
df = pd.read_csv("../input/siim-covid19-dataset-256px-jpg/train.csv")

df["image_label"] = df["image_label"].map(lambda x : x.split(" ")[0])
df["image_label_id"] = df["image_label"].map(lambda x  : image_dict[x])
df["filepath"] = df["ImageInstanceUID"].map(get_path)
df.head(3)
```

Splitting Data

```
train_df,val_df = train_test_split(df,
                                   test_size=0.2,
                                   random_state = SEED,
                                   stratify = df.image_label.values
                                   )

train_df.loc[:,"data"] = "train"
val_df.loc[:,"data"] = "val"
df = pd.concat([train_df, val_df]).reset_index(drop=True)
```

Preparing folder structure

```
os.makedirs('../siim_covid19/dataset/images/train', exist_ok=True)
os.makedirs('../siim_covid19/dataset/images/val', exist_ok=True)

os.makedirs('../siim_covid19/dataset/labels/train', exist_ok=True)
os.makedirs('../siim_covid19/dataset/labels/val', exist_ok=True)
print("Created folder structure")
```

```
IMAGE_PATH = "../siim_covid19/dataset/images"
for i in df.values:
    data = i[10]
    img_name = i[9].split("/")[-1]
    shutil.copyfile(i[9],f"{IMAGE_PATH}/{data}/{img_name}")
```

Creating YAML file

```
# REF :  https://www.kaggle.com/ayuraj/train-covid-19-
detection-using-yolov5

yaml_dict = dict(
    train = "../dataset/images/train",
    val = "../dataset/images/val",
    nc = 2,
    names = ["none","opacity"]
)

with open("../siim_covid19/yolov5/data/data.yaml", "w") as f:
    yaml.dump(yaml_dict,f,default_flow_style=True)
```

Preprocessing Bounding Boxes

```
df["boxes"] = df["boxes"].map(lambda x : ast.literal_eval(x))

def preprocess_bbox(row):
    factor_x = 1/row[5]
    factor_y = 1/row[4]
    bboxes = []

    if row[7] == "opacity":
        for box in row[6]:
            x = box["x"]*factor_x
            y = box["y"]*factor_y
            w = box["width"]*factor_x
            h = box["height"]*factor_y
            xc = x + w/2
            yc = y + h/2

            bboxes.append([xc,yc,w,h])
    return bboxes
```

Preprocessing Bounding Boxes

```
# Prepare txt files
LABEL_PATH = "../siim_covid19/dataset/labels"
for row in df.values:
    filename = row[9].split("/")[-1][:-4]
    filepath = f"{LABEL_PATH}/{row[10]}/{filename}.txt"

    if row[7] == "opacity":
        bbox = preprocess_bbox(row)
        with open(filepath, "w") as f:
            for box in bbox:
                box = [1] + box
                box = [str(i) for i in box]
                box = ' '.join(box)
                f.write(box)
                f.write('\n')
```

Training

```
%cd yolov5/
!wandb off
!python train.py --img {HEIGHT} \
            --batch {BATCH_SIZE} \
            --epochs {EPOCHS} \
            --data data.yaml \
            --weights yolov5s.pt \
            --save_period 1\
            --name yolov5_training

print("Training Completed")
```

```
%cd ../../
train_ls =
tf.io.gfile.glob("./siim_covid19/yolov5/runs/train/yolov5_traini
ng/*.*g")

os.makedirs("./working/train_results", exist_ok=True)
for filepath in train_ls:
   name = filepath.split("/")[-1]
   shutil.copyfile(filepath,"./working/train_results/"+name)

shutil.copyfile("./siim_covid19/yolov5/runs/train/yolov5_traini
ng/weights/best.pt","./working/best_yolov5.pt")
print("Training curves moved to output directory")
```

Inference

```
TEST_PATH = "/kaggle/input/siim-covid19-dataset-256px-jpg/224px/test/test"
BEST_MODEL_PATH  ="./runs/train/yolov5_training/weights/best.pt"

%cd ./siim_covid19/yolov5
!python detect.py --weights {BEST_MODEL_PATH} \
                  --source {TEST_PATH} \
                  --img {HEIGHT} \
                  --conf 0.3 \
                  --iou-thres 0.5 \
                  --max-det 3 \
                  --save-txt \
                  --save-conf \
                  --name yolov5_testing


print("Predictions Completed")
```

Format of txt file of output

label_id x_center y_center width height

Sample Prediction

```
PREDICTIONS_PATH = "runs/detect/yolov5_testing/labels/"
PRED_FILES = os.listdir(PREDICTIONS_PATH)

print("Sample prediction (in txt file) : \n")

with open(PREDICTIONS_PATH + PRED_FILES[0], "r") as f:
    ls = f.read().strip("\n").split(" ")
    print(f"LABEL : {ls[0]} \nX_CENTER : {ls[1]} \nY_CENTER : {ls[2]} \nWIDTH : {ls[3]}
\nHEIGHT : {ls[4]} \nCONFIDENCE : {ls[5]}")
print("Number of Prediction file originally present : ",len(PRED_FILES))
```

Creating txt files for "none" class

```
IMAGE_FILES = tf.io.gfile.glob("runs/detect/yolov5_testing/*.jpg")
ACTUAL_FILES = []
for path in IMAGE_FILES:
    ls = path.split("/")
    name = ls[-1].split(".")[0] + ".txt"
    ACTUAL_FILES.append(PREDICTIONS_PATH + name)

for filepath in ACTUAL_FILES:
    if not os.path.exists(filepath):
        with open(filepath,"w") as f:
            f.write("0 0 0 1 1 1")
            f.close()
```

Preprocessing functions

```
##---------------------------------------
#getting string of predictions from txt file
##---------------------------------------
def get_string(filepath,out= "bbox"):
    probs = []
    bboxes = []
    labels = []
    with open(filepath, "r") as f:
        for line in f:
            ls = line.strip("\n").split(" ")
            ls = list(map(float, ls))
            labels.append(ls[0])
            bboxes.append(ls[1:-1])
            probs.append(ls[-1])
    if out == "bbox":
        return bboxes
    elif out == "label":
        return labels
    else:
        return probs
```

```
##---------------------------------------
#Scaling-up bounding box co-ordinates
##---------------------------------------
def scale_bbox(row):
    scale_x = row[5]
    scale_y = row[6]
    scale_bboxes = []
    for box in row[1]:
        if row[2][0] != 0.0:
            xc,yc = box[0]*scale_x,box[1]*scale_y
            w,h = box[2]*scale_x,box[3]*scale_y
            xmin,ymin = int(xc - w/2),int(yc - h/2)
            xmax,ymax = int(xc + w/2),int(yc + h/2)
            scale_bboxes.append([xmin,ymin,xmax,ymax])
        else:
            return [[0,0,1,1]]
    return scale_bboxes
```

Preprocessing prediction csv file

```
pred_df = pd.DataFrame.from_dict({"filepath" : ACTUAL_FILES})
pred_df["bboxes"] = pred_df["filepath"].map(lambda x: get_string(x,out="bbox"))
pred_df["label"] = pred_df["filepath"].map(lambda x: get_string(x,out="label"))
pred_df["confidence"] = pred_df["filepath"].map(lambda x:
get_string(x,out="conf"))

pred_df["ImageInstanceUID"] = pred_df["filepath"].map(lambda x : x.split("/")[-
1].split("_")[-1][:-4])

#%cd ../
meta_df = pd.read_csv("/kaggle/input/siim-covid19-dataset-256px-
jpg/meta_test.csv")
pred_df = pred_df.merge(meta_df, on = "ImageInstanceUID")

box_ls = []
for i,row in enumerate(pred_df.values):
   box_ls.append(scale_bbox(row))

pred_df["scale_bboxes"] = box_ls

pred_df.head(3)
```

Final Study Level Predictions

Final Study Level Predictions

```
%cd ../../../
image_dict = {
    0. : "none",
    1. : "opacity"
}

def get_string(row):
    string = ""
    if row[2][0] == 0.:
        string += "none 1 0 0 1 1"
        return string
    for i in range(len(row[2])):
        string += "opacity "
        string += str(row[3][i])
        string += " "
        bbox = map(str,row[7][i])
        tmp = " ".join(bbox)
        string += tmp
        string += " "
    return string

strings = []
for row in pred_df.values:
    strings.append(get_string(row))


test_df = pd.DataFrame.from_dict({"Id" : pred_df["ImageInstanceUID"]})
test_df["Id"] = test_df['Id'].map(lambda x : x+"_image")
test_df["PredictionString"] = strings
test_df.to_csv("kaggle/working/study_level_pred.csv",index = False)
test_df.head()
```

Información yolov5

CI tests verify correct operation of YOLOv5 training (train.py), validation (val.py), inference (detect.py) and export (export.py) on MacOS, Windows, and Ubuntu every 24 hours and on every commit.

https://towardsdatascience.com/yolo-v5-object-detection-tutorial-2e607b9013ef

https://jooskorstanje.com/yolov5-training-a-custom-object-detection-model.html