Small Various Object Detection using YOLO5

Khalid Hasan (id: 17-34538-2)[a], Aninda Kumar Sharma (id: 17-35513-3)[a], Syed Fahim Intisar (id:18-36355-1)[a], Nur-A-Marzan Dipro (id: 19-39529-1)[a]

[a]*Department of Computer Sciences, American International University-Bangladesh*

Abstract

Object detection is a form of computer technology that deals with detecting instances of semantic objects of a certain class in digital images and videos. Face detection and pedestrian detection are two well-studied object detection areas. You Only Look Once (YOLO) is an algorithm proposed by Redmond et al in a research study published as a conference paper at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), which won the OpenCV People's Choice Award. Object detection methods before YOLO took a different strategy, repurposing classifiers to conduct detection. YOLO proposes using an end-to-end neural network to make bounding box and class probability predictions all at once. Following a month of the arrival of the YOLOv4 model on 27 May 2020, YOLOv5[4] got delivered by Glenn Jocher(Founder and CEO of Utralytics). It was later freely available on Github. Glenn presented the YOLOv5 in Pytorch based methodology. So, YOLOv5 is written in the Pytorch structure. YOLOv5 is considered as fast, precise and easy to train and has a long and effective history of real time detecting of objects. In this paper, we implemented the YOLOv5 architecture to detect objects. By testing this dataset, we attempted to determine the model's efficiency on detecting objects. We report our exploratory examinations on one specific dataset coco2017. Finally, detection results are made based on the results from the experimental study. Our results show that YOLOv5 proves to be a efficient state-of-the-art object detector which can perform object detection effectively.

Keywords—object detection, convolutional neural network, YOLOv5, real-time, dataset.

1. Introduction

Object detection is a computer vision strategy that allows us to differentiate and find objects in an picture or video. Object detection comprises in both assessing the precise object position which is named object localization and deciding which class it's an area with which is named object classification. It's a focal segment in numerous genuine applications, identification of location of individuals on foot and vehicles for self-governing driving, detection of people to research their practices, location, and further acknowledgment of face recognition for observation purposes, and so on. In any case, due to the various sorts of perspectives, postures, impediment, and lighting conditions, object

detection stays a difficult issue. Before the approaching of profound machine learning methods, the quality object detection technique in computer vision was made from three principal steps. The primary step was selection of districts of interest (ROIs), where objects may show up or be available. Next step was removing a couple of attributes from these districts/objects (a.k.a. highlight extraction). Last step was taking care of an administered classifier with these components. These supposed handmade highlight configuration-based methodologies have commonly given great outcomes to continuous applications in functional frameworks [1]. However, they were for the foremost part unfit to surpass a selected degree of exactness. Inside the last decade, the arising advancement of profound deep neural organizations, and more especially of convolutional neural networks (CNNs), brought both a worldview changes and a few critical enhancements [2,3]. This addition in precision has been conceivable, thanks not exclusively to some methodological advances, yet additionally to the accessibility of tons of data and to the expanded computing capacity.

There have been many techniques in this field from various perspectives. Diverse techniques have been used in this field. The design of CNN networks, loss functions, data augmentations, and training strategies are all possible research directions. YOLOv4[23] explores all of these research directions and proposes the YOLOV4 object detector, which is based on network architecture optimizations. Then came YOLOv5[4]. YOLOv5 is a group of compound-scaled object recognition models prepared on the COCO dataset, and incorporates basic usefulness for Test Time Augmentation (TTA), model ensembling, hyperparameter advancement, and fare to ONNX, CoreML and TFLite. Now using the YOLOv5 object detector, we implement a object detector based on that intuition. Ultimately, we want to provide a variety of models for different applications ranging from the highly complex to the simple. In this paper, we intend to implement YOLOv5 architecture. We tried to investigated the robustness of the architecture. We have selected coco2017 dataset to test the model. In the rest of the paper, Part 2 momentarily goes through writing works in object location in far off detecting with an emphasis on little object detection. Segment 3 discusses about methodology of YOLO's object detection technique and portrays architecture of YOLOv5. In Part 4, we report our exploratory examinations on the selected dataset. In the last part, the result and improvement possibilities are considered and discussed.

2. Literature Review

Numerous endeavors are made inside the computer vision area to manage the recognition of little objects, essentially by adjusting the predominant one-stage and two-stage systems. Various new works have taken advantage of cutting edge two-stage locators, as Faster R-CNN [12,13], deconvolution R-CNN [14], and deformable R-CNN

[15,16] to distinguish e.g., little vehicles, planes, ships, man-made designs, in far off detecting datasets gathered for the most part from aeronautical or Google Earth pictures. In another survey, Some creators additionally featured some incredible models for distinguishing nonexclusive little objects, as further developed Faster R-CNN [6,7], Feature-combined SSD [8], and MDSSD (Mutual Detection of Small Objects) [9], RefineDet [10], SCAN (Semantic setting mindful organization) [11], and so on The recognition of little object s has been for the most part handled by taking advantage of two-stage object detectors due to their ability to by and large give more exact identification execution inside the distant detecting local area.

They additionally featured some incredible models to identify nonexclusive little objects, as further developed Faster R-CNN [6,7], Feature-melded SSD [8], MDSSD (Multi-scale deconvolutional SSD) [9], RefineDet [10], SCAN (Semantic setting mindful organization) [11], and so on. Inside the far off detecting local area, the identification of little objects has been for the most part handled by taking advantage of two-stage object indicators due to their ability to by and large give more exact recognition execution when contrasted with one-stage locators. During this situation, the area-based methodology gives off an impression of being more important since the essential stage whose point is to look up-and-comer objects may be set to have practical experience in little districts and overlook enormous ones. In any, two or three examinations are proposed, explicitly utilizing YOLO-based systems to recognize little object s from far off detecting pictures, similar to individuals [17], airplanes [18], ships [19], and building impressions [20] . Because of their lower location precision contrasted with district based locators, YOLO indicators have drawn in a relatively bit of interest. UAV-YOLO was proposed in [17] as how to adjust the YOLOv3 to distinguish little objects from automated airborne vehicle (UAV) information. Because of their lower recognition precision contrasted with locale based identifiers, YOLO detectors aren't the essential decision for continuous COMPUTER vision applications. This is regularly especially valid for little object s. For example , to identify little object s from automated aeronautical vehicle (UAV) information, UAV-YOLO was proposed [17]. YOLOv3 was marginally adjusted by connecting two indistinguishable estimated leftover squares of the organization backbone. At the point when YOLOv3 was utilized in [18], the creators showed that its computational presentation was better than that of Faster R-CNN and SSD. YOLOv3 and YOLT (You Only Look Twice) [21], which is very like YOLOv2, were analyzed in [19] for transport identification. By and by, no progressions to the first design were proposed. In early 2020, YOLOv4[23] was released. It was released by a different research team in early 2020, after the original YOLO authors had left the research field. After one month, the YOLOv5 [4] was delivered by another distinctive research group. In the calculation planned, the YOLOv5 [4] doesn't have numerous developments. However, the group doesn't distribute a paper. These bring very a few discussions about in the event that it ought to be called YOLOv5. Be that as it may, because of its fundamentally diminished model size, quicker speed, and comparative per- formance as YOLOv4 [23], and a full execution in Python (Pytorch), it is welcome by the object recognition local area.

3. Methodology

YOLO network has 24 convolutional layers followed by 2 completely related layers. YOLO utilize 1 × 1 reductive layers followed by 3 × 3 convolutional layers instead of the starting modules utilized by GoogLeNet,. The input image is divided into a SxS grid by the system. If the object's center falls inside a grid cell, that grid cell is in charge of detecting the object. B bounding boxes and confidence scores for those boxes are predicted in each grid cell. These confidence scores represent the model's belief that the box contains an object as well as the accuracy with which it believes the box it forecasts is. Confidence is defined as Pr(Object) IOU in formal terms. The confidence ratings should be zero if no object exists in that cell. Otherwise, YOLO wants the confidence score to be equal to the IOU between the forecast box and the ground reality. There are five predictions in each bounding box: x,confidence, y, w, and h. The (x, y) coordinates represent the box's centroid with relation to the grid cell's limits. The width and height are calculated in terms of the entire image. Finally, the IOU between the projected box and any ground truth box is represented by the confidence prediction. Each grid cell additionally forecasts Pr(Classi |Object), which are C conditional class probabilities. These probabilities are based on the grid cell in which an object is located. Regardless of the number of boxes B, YOLO only predicts one set of class probabilities per grid cell. YOLO multiplies the conditional class probabilities and individual box confidence forecasts at test time, giving us class-specific confidence predictions. These confidence ratings represent the likelihood of that class being in the box as well as the accuracy with which the projected box matches the object. [22]

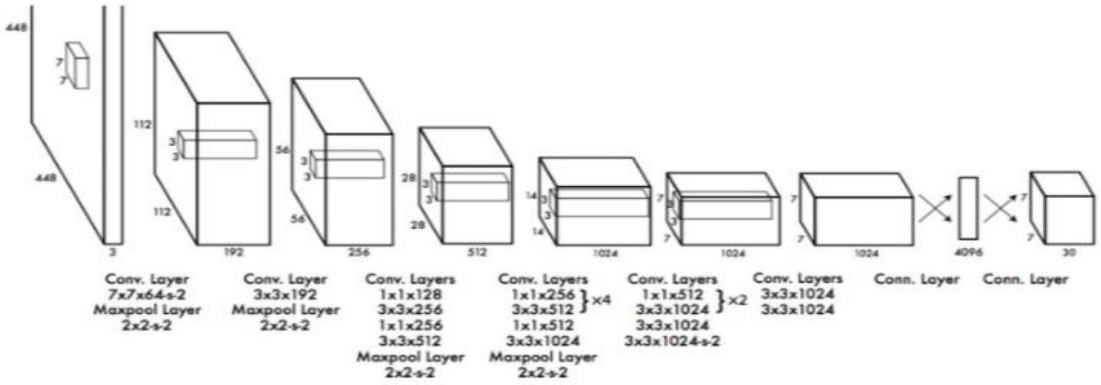


Figure 1: YOLO (You Only Look Once) Architecture

Stood apart from YOLOv3, different changes have been made in YOLO model. One of them is YOLOv5 which is also a state-of-the-art object detector that can detect objects in real-time. On two official object detection datasets: Pascal VOC (visual objects classes)

[24] and Microsoft COCO (common objects in context)[25], it has consistently outperformed the competition. Figure 2 shows the network architecture of Yolov5. We chose Yolov5 for three reasons. CSPNet (cross stage partial network) [26] was incorporated by Yolov5 into Darknet, creating CSPDarknet as the backbone. In large-scale backbones, CSPNet solves the problem of repeated gradient information, and integrates the gradient changes into the feature map, resulting in an improved feature map. It also used a path aggregation network (PANet) [28] as a neck in order to improve information flow. It uses a new feature pyramid network (FPN) structure with an improved bottom-up path, which improves the propagation of low-level features in PANet's architecture. It is also possible to propagate useful information from one subnetwork to the next by using an adaptive feature pooling technique, which links the feature grid and all feature levels. Using PANet, Lowe can better utilize accurate localization signals. As a result of PANet, more accurate localization signals can be used in lower layers, resulting in an increase in object location accuracy. Yolov5's Yolo layer also generates three different feature maps of different sizes (18 x 18, 36 x 36, and 72 x 72 pixels), allowing the model to handle small, medium and large objects. In most cases, a forest fire progresses from a small fire (ground fire) to a medium-sized fire (trunk fire) and finally to a large-scale fire (treefire) (canopy fire). As a result of multi-scale detection, the model is able to track size changes in the fire evolution process. Additionally, repeated gradient information in large-scale networks is solved by CSPNet. Speed and accuracy are critical in our detection task, and a small model size determines its inference efficiency on resource-limited edge devices.[27]

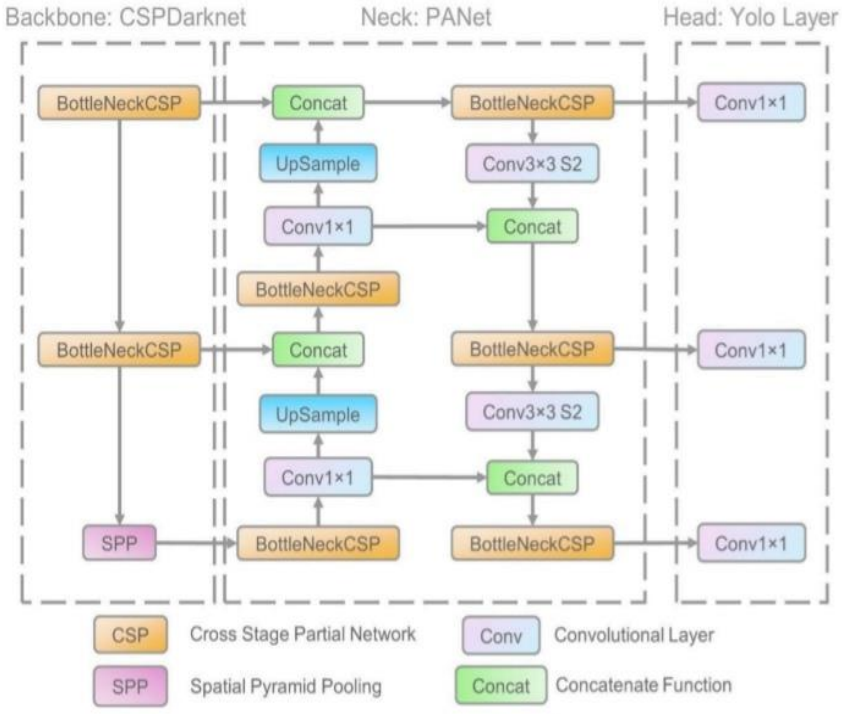


Figure 2: YOLOv5 Architecture [27]

4. Experimental Study

YOLO "**You Only Look Once**" is one of the most popular and most favorite algorithms for AI engineers. It always has been the first preference for real-time object detection.YOLO has emerged so far since it's the first release. Let's briefly discuss earlier versions of YOLO then we will jump straight into the training part.

4.1 Previous YOLO versions: YOLO v1 was introduced in 2016 by Joseph Redmon et al with a research paper called "You Only Look Once: Unified, Real-Time Object Detection". This was the initial paper by Redmon that revolutionized the industry and changed the Real-Time Object detection methods totally. By just looking at the image once, it can detect the objects with a speed of 45fps (frames per second), another YOLO v1 type, **Fast YOLOv1** was able to achieve 155fps with little less accuracy.

It used the Darknet framework that was trained on the ImageNet-1000 dataset. But YOLOv1 has many limitations like

- it can't detect the objects properly when the objects are small
- it also can't generalize the objects if the image is of different dimensions
  YOLOv2(YOLO9000)

The second version of YOLOv2 was released in 2017 by Ali Farhadi and Joseph Redmon. This time Joseph collaborated with Ali for major bug fixes and accuracy increment. The research they published was **"YOLO9000: Better, Faster, Stronger."** The name of the second version of YOLO was *YOLO9000*. The major competitor of YOLO9000 was **Faster R-CNN,** which was also an object detection algorithm that uses **Region Proposal Network** & **(SSD)Single-shot Multbox Detector** to identify the multiple objects from an image.

Some of the features of YOLOv2 are:

- YOLOv2 added **Batch Normalization** as an improvement that normalizes the input layer of the image by altering the activation functions.
- Higher-resolution input: input size has been increased from 224*224 to 448*448.
- Anchor boxes.
- Multi-Scale training.
- Darknet 19 architecture with **19 convolution layers** and **5 Max Pooling layers**.
  YOLOv2 performance on MS COCO dataset

4 2 YOLOv3:

After one year, on March 25, Joseph Redmon and Ali Farhadi came up with another version of YOLO and a research paper called: **"YOLOv3: An Incremental improvement."**

At 320×320, YOLOv3 runs with 22ms at 28.2 mAP with great accuracy, as shown in the above video. It is three times faster than the previous **SSD** and four times faster than RetinaNet.

New YOLOv3 followed the methodology of the previous YOLOv2 version: YOLO9000. In this approach, Redmond uses **Darknet 53** architecture, which was a significantly improved version and had 53 convolution layers.

Some of the new, improved features in YOLOv3 was:

- Class Predictions
- Feature Pyramid Networks (FPN)
- Darknet 53 architecture

4.3 YOLOv4:

As Redmond was not currently working on the CV for a long time, a new team of three developers released YOLOv4. It was released by Alexey Bochoknovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Alexey is the one who developed the Windows version of YOLO back in the days.

YOLOv4 runs twice faster than EfficientDet with comparable performance, as shown in the below diagram, which was officially published on the YOLOv4 research paper.

Some of the new features of YOLOv4 is:

- Anyone with a 1080 Ti or 2080 ti GPU can run the YOLOv4 model easily.
- YOLOv4 includes CBN (Cross-iteration batch normalization) and PAN (Pan aggregation network) methods.
- Weighted-Residual-Connections (WRC).
- Cross-Stage-Partial connections (CSP), a new backbone to enhance CNN (convolution neural network)

- Self-adversarial-training (SAT): A new data augmentation technique
- DropBlock regularization.

4.4 YOLOv5:

After a few days of the release of the YOLOv4 model on 27 May 2020, YOLOv5 got released by Glenn Jocher(Founder & CEO of Utralytics). It was publicly released on GitHub here. Glenn introduced the YOLOv5 Pytorch based approach, and yes! YOLOv5 is written in the Pytorch framework. It is state of the art and newest version of the YOLO object detection series, and with the continuous effort and 58 open-source contributors, YOLOv5 set the benchmark for object detection models very high; as shown below, it already beats the EfficientDet and its other previous YOLOv5 versions. There is no official paper released yet and also many controversies are happening about its name. Now Let's see some coding examples that was published with its code at GitHub for learning purposes.

Pytorch inferences are very fast that before releasing YOLOv5, many other AI practitioners often translate the YOLOv3 and YOLOv4 weights into Ultralytics **Pytorch** weight.

Implementation

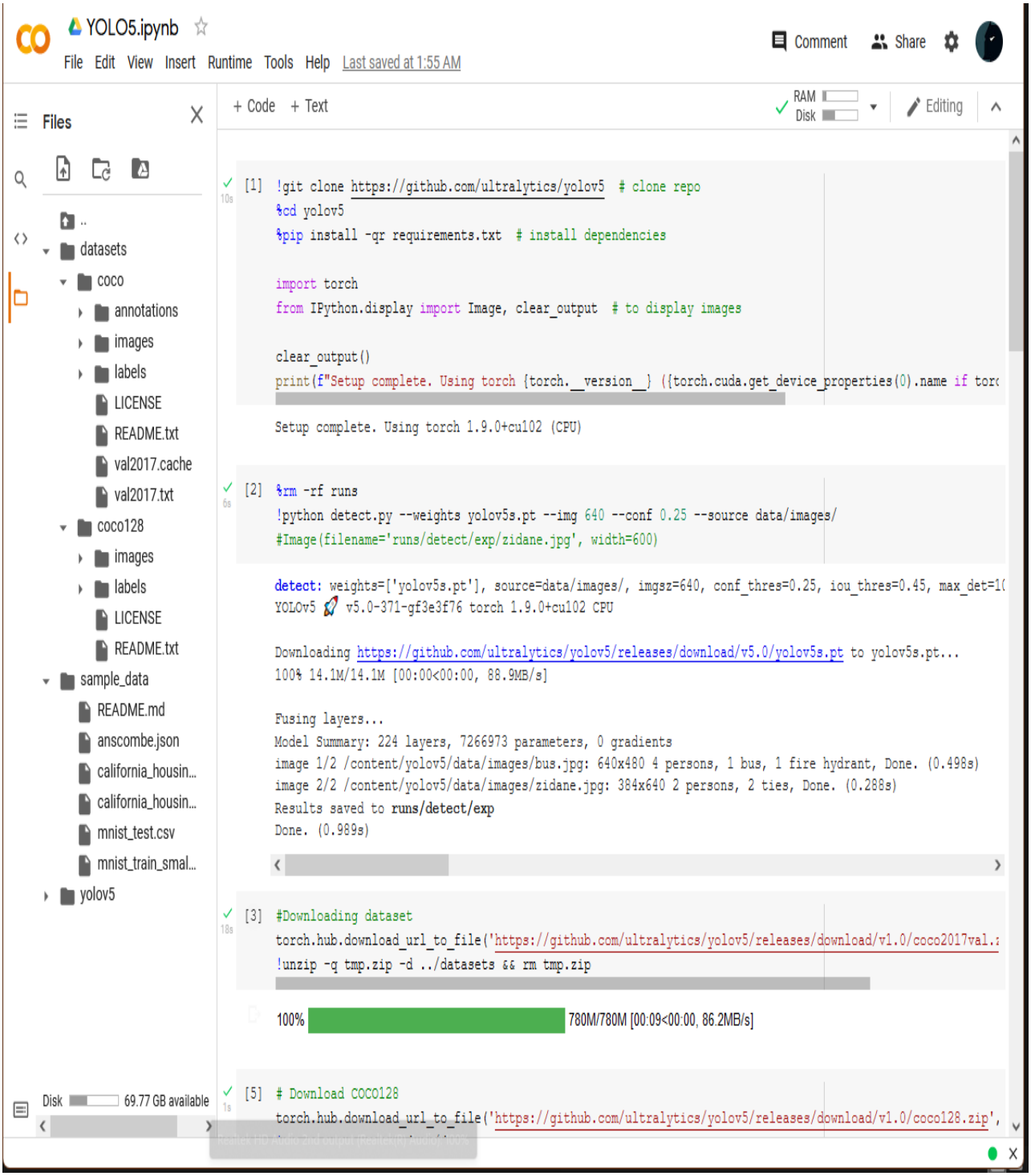
We are going to implement starter tutorial on YOLOv5 by Ultralytics and going to detect some objects from our given image.

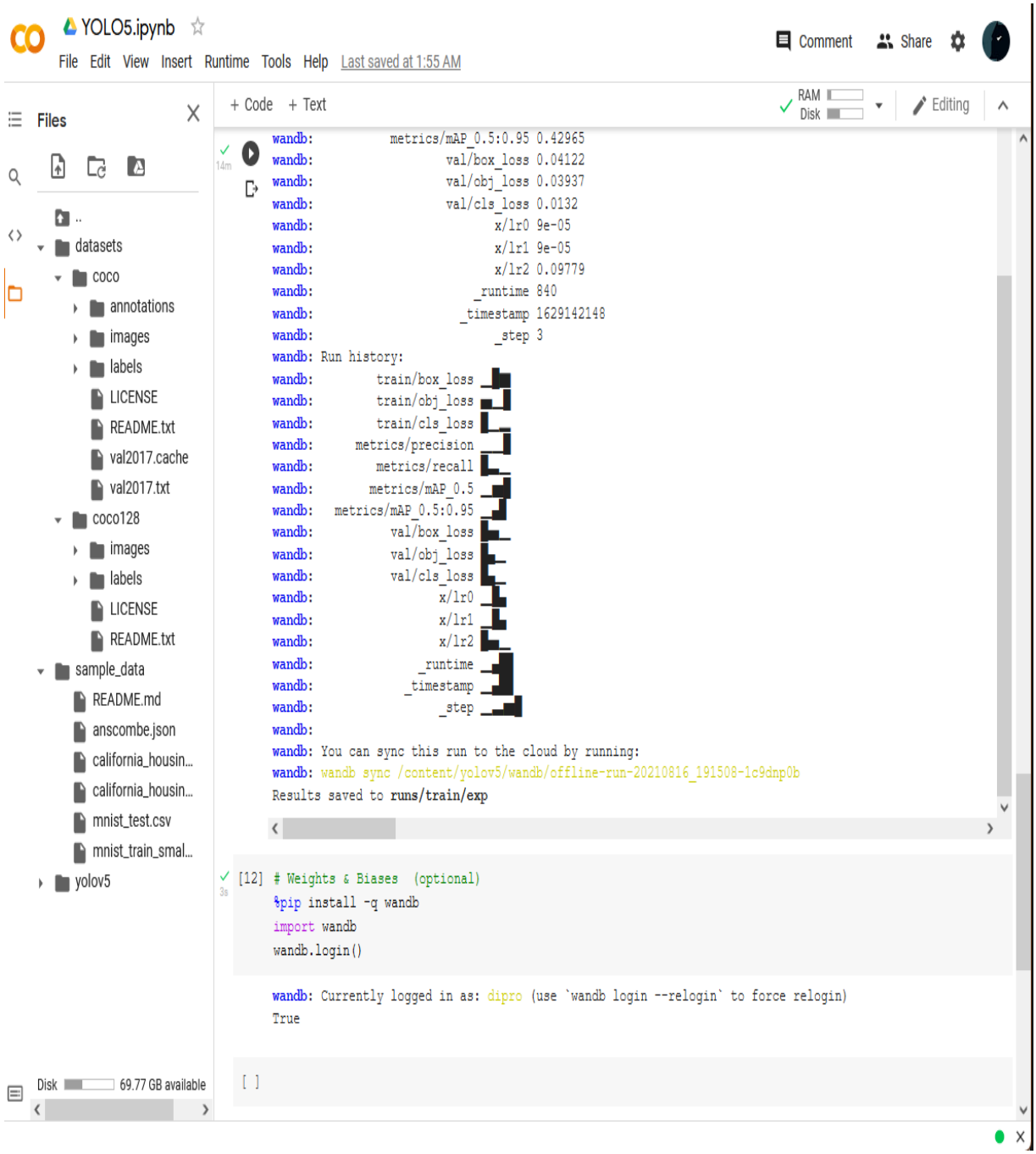
1. First, we clone the YOLOv5 repo from GitHub to our Google colab environment using the below command.
2. We Install the dependencies using the pip command
3. Import some of the modules like a torch and display to display our output image inside the notebook.
4. Download custom image for testing

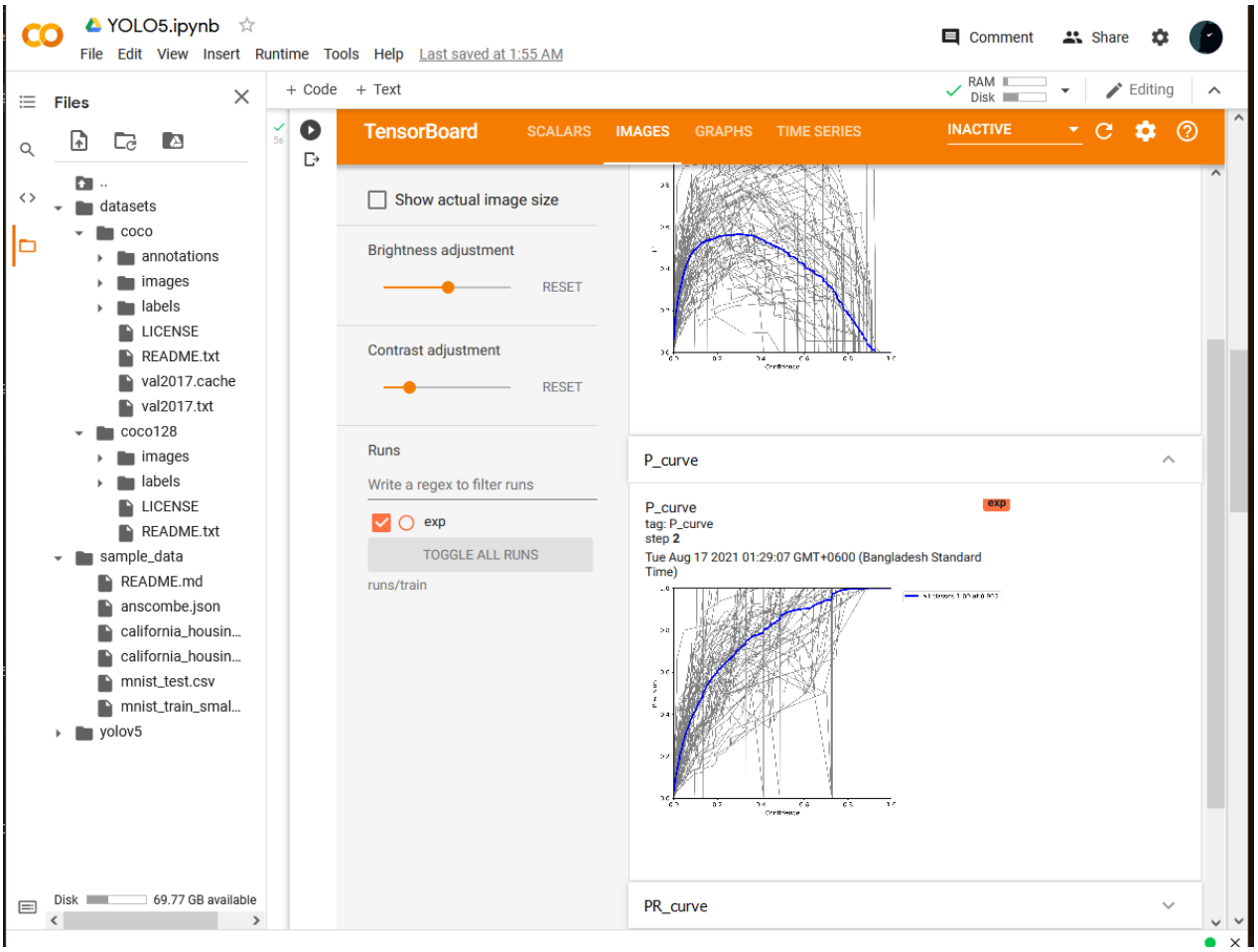
5.Test using command, **detect.py** runs inference on a variety of sources and will automatically download the latest model from.

These screen shorts slow how we implement the yolo v5 for detecting odjecr.


YOLO5.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:55 AM
Comment
Share
Files
+ Code + Text
RAM
Disk
Editing
Using 2 dataloader workers
Logging results to runs/train/exp
Starting training for 3 epochs...
Epoch gpu_mem box obj cls labels img_size
0/2 0G 0.04423 0.06615 0.02097 243 640: 100% 8/8 [03:38<00:00, 27.27s/it]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 4/4 [01:01<
all 128 929 0.694 0.568 0.642 0.42
Epoch gpu_mem box obj cls labels img_size
1/2 0G 0.04611 0.06131 0.02026 180 640: 100% 8/8 [03:20<00:00, 25.06s/it]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 4/4 [01:02<
all 128 929 0.694 0.565 0.655 0.424
Epoch gpu_mem box obj cls labels img_size
2/2 0G 0.04595 0.06889 0.02033 229 640: 100% 8/8 [03:25<00:00, 25.74s/it]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 4/4 [01:02<
all 128 929 0.712 0.564 0.66 0.43
3 epochs completed in 0.228 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.8MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.8MB
wandb: Waiting for W&B process to finish, PID 381
wandb: Program ended successfully.
wandb: Find user logs for this run at: /content/yolov5/wandb/offline-run-20210816_191508-1c9dnp0b/logs/d
wandb: Find internal logs for this run at: /content/yolov5/wandb/offline-run-20210816_191508-1c9dnp0b/lo
wandb: Run summary:
wandb: train/box_loss 0.04595
wandb: train/obj_loss 0.06889
wandb: train/cls_loss 0.02033
wandb: metrics/precision 0.71221
wandb: metrics/recall 0.56366
wandb: metrics/mAP_0.5 0.66029
wandb: metrics/mAP_0.5:0.95 0.42965
wandb: val/box_loss 0.04122
wandb: val/obj_loss 0.03937
wandb: val/cls_loss 0.0132
wandb: x/lr0 9e-05
wandb: x/lr1 9e-05
wandb: x/lr2 0.09779
wandb: _runtime 840
wandb: _timestamp 1629142148
wandb: _step 3
wandb: Run history:
..
datasets
coco
annotations
images
labels
LICENSE
README.txt
val2017.cache
val2017.txt
coco128
images
labels
LICENSE
README.txt
sample_data
README.md
anscombe.json
california_housin...
california_housin...
mnist_test.csv
mnist_train_smal...
yolov5
Disk 69.77 GB available


YOLO5.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:55 AM
Comment
Share
Files
+ Code + Text
RAM
Disk
Editing
datasets
coco
annotations
images
labels
LICENSE
README.txt
val2017.cache
val2017.txt
coco128
images
labels
LICENSE
README.txt
sample_data
README.md
anscombe.json
california_housin...
california_housin...
mnist_test.csv
mnist_train_smal...
yolov5
Disk 69.77 GB available
[1] !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 torc
Setup complete. Using torch 1.9.0+cu102 (CPU)
[2] %rm -rf runs
!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images/
#Image(filename='runs/detect/exp/zidane.jpg', width=600)
detect: weights=['yolov5s.pt'], source=data/images/, imgsz=640, conf_thres=0.25, iou_thres=0.45, max_det=1(
YOLOv5 v5.0-371-gf3e3f76 torch 1.9.0+cu102 CPU
Downloading https://github.com/ultralytics/yolov5/releases/download/v5.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 88.9MB/s]
Fusing layers...
Model Summary: 224 layers, 7266973 parameters, 0 gradients
image 1/2 /content/yolov5/data/images/bus.jpg: 640x480 4 persons, 1 bus, 1 fire hydrant, Done. (0.498s)
image 2/2 /content/yolov5/data/images/zidane.jpg: 384x640 2 persons, 2 ties, Done. (0.288s)
Results saved to runs/detect/exp
Done. (0.989s)
[3] #Downloading dataset
torch.hub.download_url_to_file('https://github.com/ultralytics/yolov5/releases/download/v1.0/coco2017val.
!unzip -q tmp.zip -d ../datasets && rm tmp.zip
100% 780M/780M [00:09<00:00, 86.2MB/s]
[5] # Download COCO128
torch.hub.download_url_to_file('https://github.com/ultralytics/yolov5/releases/download/v1.0/coco128.zip',

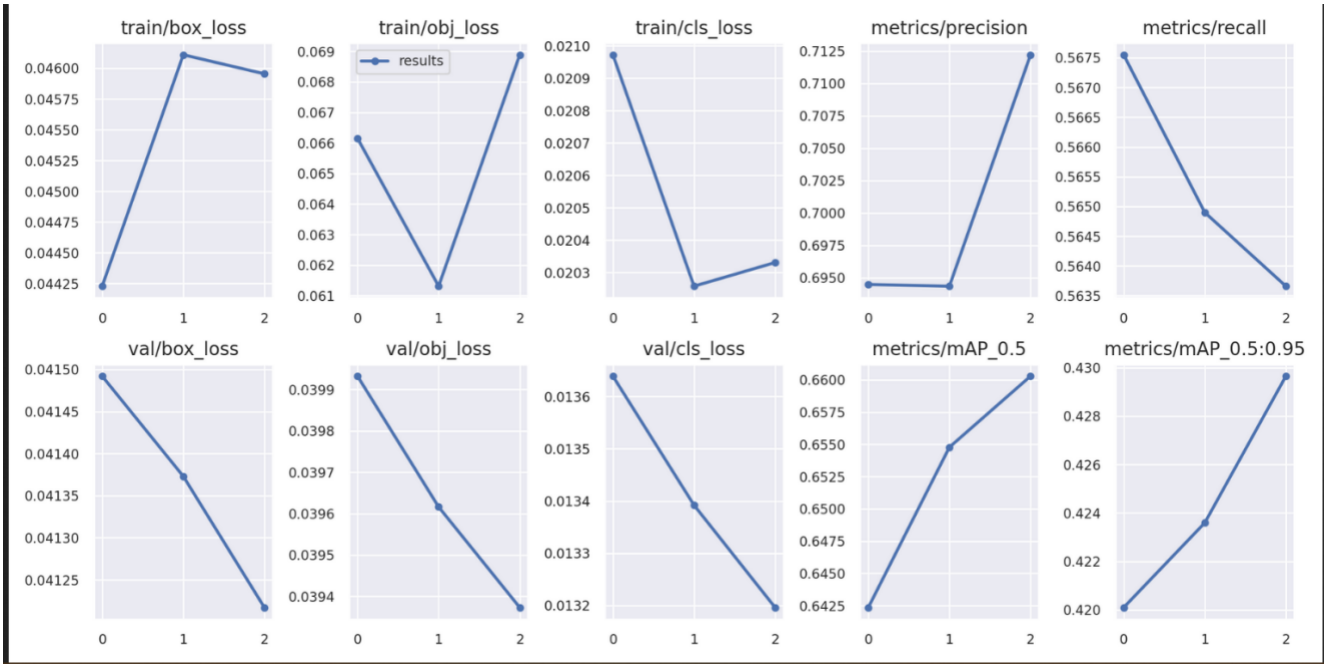

YOLO5.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:55 AM
Comment
Share
Files
+ Code + Text
RAM
Disk
Editing
wandb: metrics/mAP_0.5:0.95 0.42965
wandb: val/box_loss 0.04122
wandb: val/obj_loss 0.03937
wandb: val/cls_loss 0.0132
wandb: x/lr0 9e-05
wandb: x/lr1 9e-05
wandb: x/lr2 0.09779
wandb: _runtime 840
wandb: _timestamp 1629142148
wandb: _step 3
wandb: Run history:
wandb: train/box_loss
wandb: train/obj_loss
wandb: train/cls_loss
wandb: metrics/precision
wandb: metrics/recall
wandb: metrics/mAP_0.5
wandb: metrics/mAP_0.5:0.95
wandb: val/box_loss
wandb: val/obj_loss
wandb: val/cls_loss
wandb: x/lr0
wandb: x/lr1
wandb: x/lr2
wandb: _runtime
wandb: _timestamp
wandb: _step
wandb:
wandb: You can sync this run to the cloud by running:
wandb: wandb sync /content/yolov5/wandb/offline-run-20210816_191508-1c9dnp0b
Results saved to runs/train/exp
[12] # Weights & Biases (optional)
%pip install -q wandb
import wandb
wandb.login()
wandb: Currently logged in as: dipro (use `wandb login --relogin` to force relogin)
True
[ ]
datasets
coco
annotations
images
labels
LICENSE
README.txt
val2017.cache
val2017.txt
coco128
images
labels
LICENSE
README.txt
sample_data
README.md
anscombe.json
california_housin...
california_housin...
mnist_test.csv
mnist_train_smal...
yolov5
Disk 69.77 GB available



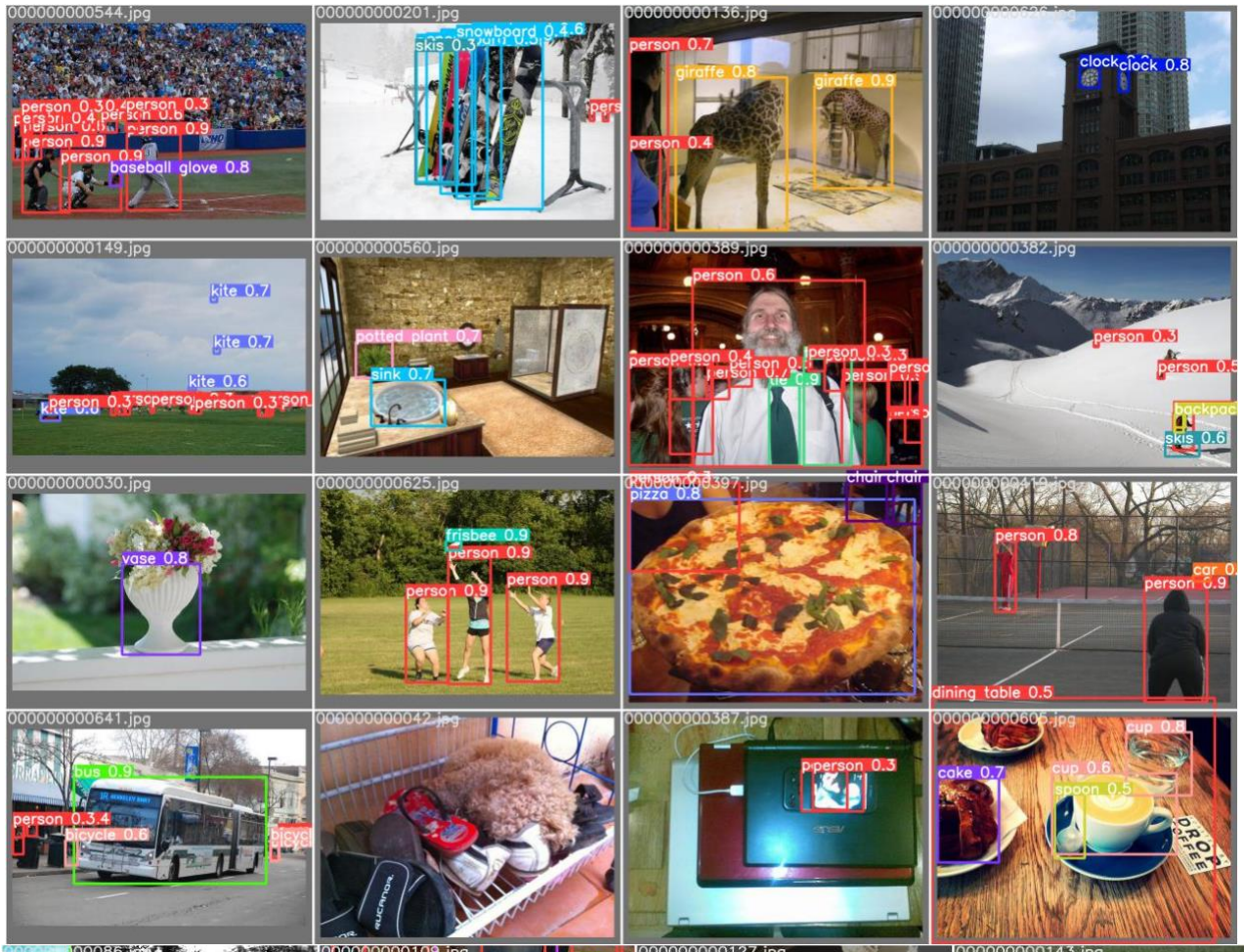
4. Result

We have defined these experiments to evaluate the results of
our proposed methodology. The YOLOv5 model has been trained with real images dataset.
Small objects in front of realistic backgrounds. Experiment
serves as a baseline that enables us to compare the training performance of a small dataset of real images to that
of a large dataset of synthetic images.

The results are shown in the following images:


train/box_loss
train/obj_loss
results
train/cls_loss
metrics/precision
metrics/recall
val/box_loss
val/obj_loss
val/cls_loss
metrics/mAP_0.5
metrics/mAP_0.5:0.95


000000298251.jpg
000000375469.jpg
000000096549.jpg
000000504000.jpg
000000205333.jpg
000000460682.jpg
000000527220.jpg
000000044590.jpg
000000449661.jpg
000000185473.jpg
000000161044.jpg
000000554266.jpg
000000485237.jpg
000000143068.jpg
000000490413.jpg


0000000000544.jpg
person 0.9
baseball glove 0.8
0000000000201.jpg
snowboard 0.6
skis 0.3
0000000000136.jpg
person 0.7
giraffe 0.8
giraffe 0.9
person 0.4
clock 0.8
0000000000149.jpg
kite 0.7
kite 0.7
kite 0.6
0000000000560.jpg
potted plant 0.7
sink 0.7
0000000000389.jpg
person 0.6
0000000000382.jpg
person 0.3
person 0.5
skis 0.6
0000000000030.jpg
vase 0.8
0000000000625.jpg
frisbee 0.9
person 0.9
person 0.9
person 0.9
pizza 0.8
chair
person 0.8
person 0.9
dining table 0.5
0000000000641.jpg
bus 0.9
person 0.3
bicycle 0.6
0000000000387.jpg
cup 0.8
cake 0.7
cup 0.6
spoon 0.5
0000000000109.jpg
0000000000127.jpg
0000000000143.jpg
0000000000110.jpg
0000000000144.jpg
0000000000113.jpg
0000000000149.jpg
0000000000094.jpg
0000000000133.jpg

5. Conclusion

In this paper, we have shown that the task of detecting small objects can be made easier, at least in natural scenarios, if we leverage the specific circumstances of this domain: Since the context is usually clearly defined and well-structured, we can make use of it in the object detection process by first recognizing the context of a small object and then the object itself. Furthermore, COCO128 dataset of small objects is usually available and can be used to generate comparatively large amounts of high-quality training data. Our proposed small object detection pipeline performs significantly better than another staged counterpart. Future work includes displaying the object detection results as holograms on the HoloLens, modelling objects and contexts in knowledge graphs for the object detection process, and leveraging eye-tracking data for small object detection.

References

1.Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. arXiv 2019, arXiv:1905.05055.

2.Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection : A survey. Int. J. Comput. Vis. 2020, 128, 261–318. [CrossRef]

3.Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z.; Qu, R. A survey of deep learning-based object detection. IEEE Access 2019, 7, 128837–128868. [CrossRef]

4.YOLOv5, “Yolov5,” .

5.Chen, F.; Ren, R.; Van de Voorde, T.; Xu, W.; Zhou, G.; Zhou, Y. Fast automatic airport detection in remote sensing images using convolutional neural networks. Remote Sens. 2018, 10, 443.

6.Eggert, C.; Brehm, S.; Winschel, A.; Zecha, D.; Lienhart, R. A closer look: Small object detection in faster R-CNN. In Proceedings of the 2017 IEEE International Conference on Multimedia and Expo (ICME), HongKong, China, 10–14 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 421–426.

7. Cao, C.; Wang, B.; Zhang, W.; Zeng, X.; Yan, X.; Feng, Z.; Liu, Y.; Wu, Z. An improved faster R-CNN for Small object detection. IEEE Access 2019, 7, 106838–106846. [CrossRef]

8. Cao, G.; Xie, X.; Yang, W.; Liao, Q.; Shi, G.; Wu, J. Feature-fused SSD: Fast detection for small objects. In Proceedings of the Ninth International Conference on Graphic and Image Processing (ICGIP 2017), Qingdao, China, 14–16 October 2017; International Society for Optics and Photonics: Bellingham, WA, USA, 2018; Volume 10615, p. 106151E.

9. Cui, L.; Ma, R.; Lv, P.; Jiang, X.; Gao, Z.; Zhou, B.; Xu, M. Mdssd: Multi-scale deconvolutional single shot Detector for small objects. arXiv 2018,

arXiv:1805.07009.

10. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-shot refinement neural network for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT,USA, 18–22 June 2018; pp. 4203–4212.

11. Guan, L.; Wu, Y.; Zhao, J. Scan: Semantic context aware network for accurate small object detection. Int. J. Comput. Intell. Syst. 2018, 11, 951–961. [CrossRef]

12. Ren, Y.; Zhu, C.; Xiao, S. Small object detection in optical remote sensing images via modified faster R-CNN. Appl. Sci. 2018, 8, 813. [CrossRef]

13.Zhang, S.; He, G.; Chen, H.B.; Jing, N.; Wang, Q. Scale adaptive proposal network for object detection in Remote sensing images. IEEE Geosci. Remote Sens. Lett. 2019, 16, 864–868. [CrossRef]

14. Zhang, W.; Wang, S.; Thachan, S.; Chen, J.; Qian, Y. Deconv R-CNN for small object detection on remote sensing images. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Valencia, Spain, 22–27 July 2018; pp. 2483–2486.

15. Ren, Y.; Zhu, C.; Xiao, S. Deformable Faster R-CNN with aggregating multi-layer features for partially occluded object detection in optical remote sensing images. Remote Sens. 2018, 10, 1470. [CrossRef]

16. Yan, J.; Wang, H.; Yan, M.; Diao, W.; Sun, X.; Li, H. IoU-adaptive deformable R-CNN: Make full use of IoU for multi-class object detection in remote sensing imagery. Remote Sens. 2019, 11, 286. [CrossRef]

17. Liu, M.; Wang, X.; Zhou, A.; Fu, X.; Ma, Y.; Piao, C. UAV-YOLO: Small Object Detection on Unmanned Aerial Vehicle Perspective. Sensors 2020, 20, 2238. [CrossRef]

18. Zhao, K.; Ren, X. Small aircraft detection in remote sensing images based on YOLOv3. In Proceedings of the International Conference on Electrical Engineering, Control and Robotics (EECR), Guangzhou, China, 12–14 January 2019; Volume 533.

19. Nina, W.; Condori, W.; Machaca, V.; Villegas, J.; Castro, E. Small ship detection on optical satellite imagery With YOLO and YOLT. In Proceedings of the Future of Information and Communication Conference (FICC), San Francisco, CA, USA, 5–6 March 2020; pp. 664–677.

20.Xie, Y.; Cai, J.; Bhojwani, R.; Shekhar, S.; Knight, J. A locally-constrained YOLO framework for detecting Small and densely-distributed building footprints. Int. J. Geogr. Inf. Sci. 2020, 34, 777–801. [CrossRef]

21. Van Etten, A. You Only Look Twice: Rapid multi-scale object detection in satellite imagery. arXiv 2018, arXiv:1805.09512.

22. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788

23. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” ArXiv preprint 2004.10934, 2020.

24. Everingham, M.; Eslami, S.A.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. Int. J. Comput. Vis. 2015, 111, 98–136, doi:10.1007/s11263-014-0733-5.

25. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the 13th European Conference on Computer Cision (ECCV 2014), Zurich, Switzerland, 6–12 September 2014; pp. 740–755.

26. Wang, C.Y.; Mark Liao, H.Y.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A new backbone that can enhance learning capability of cnn. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2020), Washington, DC, USA, 14–19 June 2020; pp. 390–391.

27. Xu, Renjie & Lin, Haifeng & Lu, Kangjie & Cao, Lin & Liu, Yunfei. (2021). A Forest Fire Detection System Based on Ensemble Learning. Forests. 12. 217. 10.3390/f12020217.

28. Wang, K.; Liew, J.H.; Zou, Y.; Zhou, D.; Feng, J. Panet: Few-shot image semantic segmentation with prototype alignment. In Proceedings of the IEEE International Conference on Computer Vision (ICCV 2019), Seoul, Korea, 20–26 October 2019; pp. 9197– 9206.