THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Final Project Group 7 Object Detection Using YOLO

Hao Heng, Ze Gong, Junhe Zhang

Introduction

- Darkent
- YOLO
- Create custom dataset
- Train YOLO network
- Result and Conclusion

What is darknet?

- Darknet is an open source neural network framework written in C and CUDA
- It is fast, easy to install, and supports CPU and GPU computation

What is Yolo?

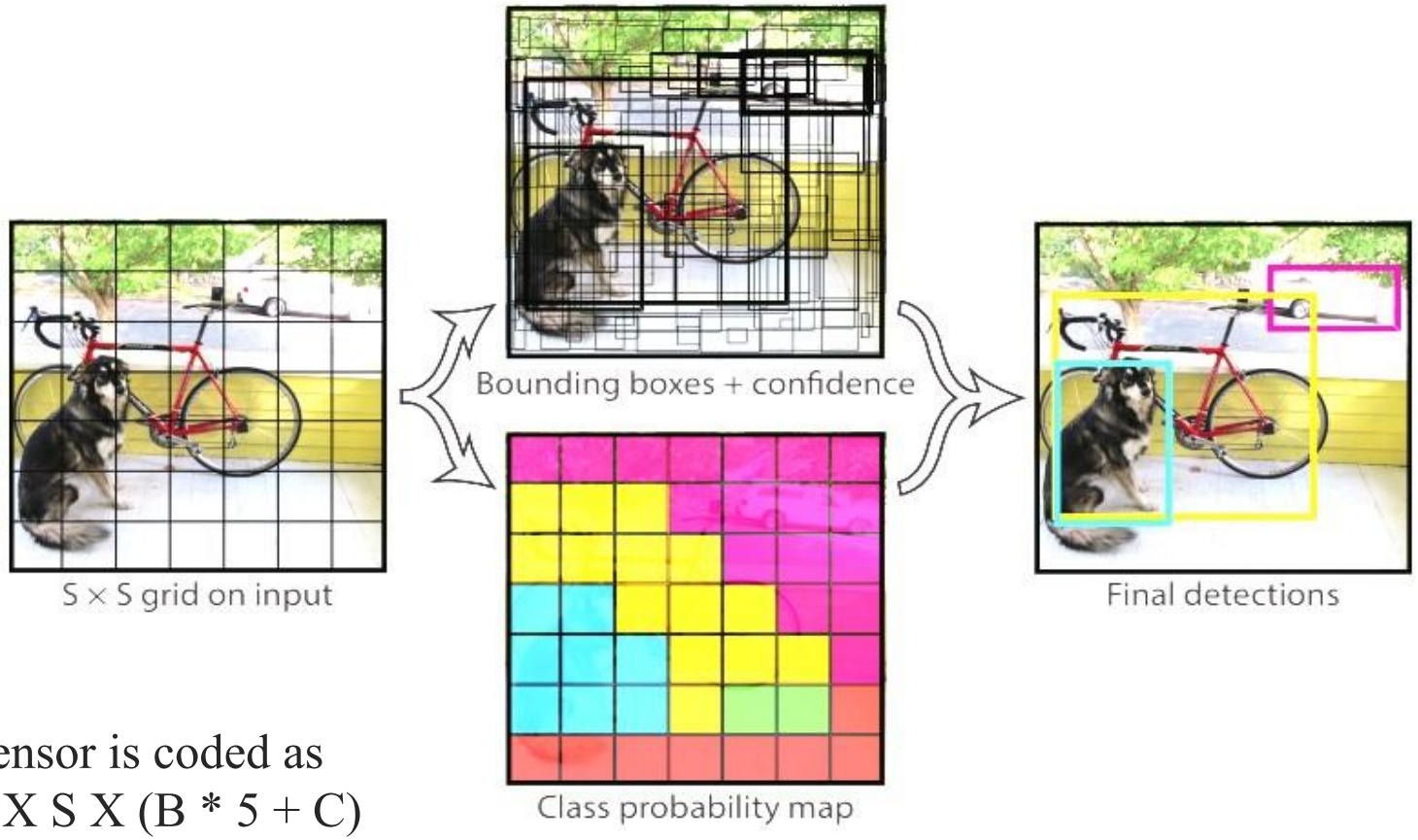
- You only look once (YOLO) is a state-of-the-art, real-time object detection system

YOLO Network

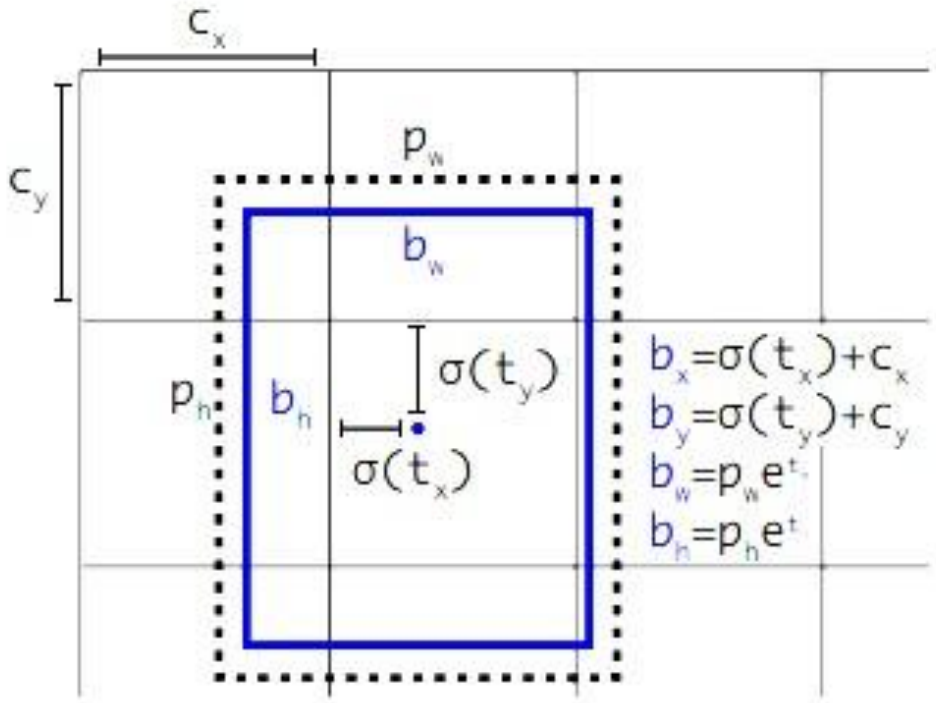
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

In this project we use YOLO (You Only Look Once) v3.0 for the purpose of Object Detection.

YOLO



Tensor is coded as
S X S X (B * 5 + C)



Calculation of the bounding box coordinates.

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Create custom dataset

Dataset

Source: Google Open Image Dataset

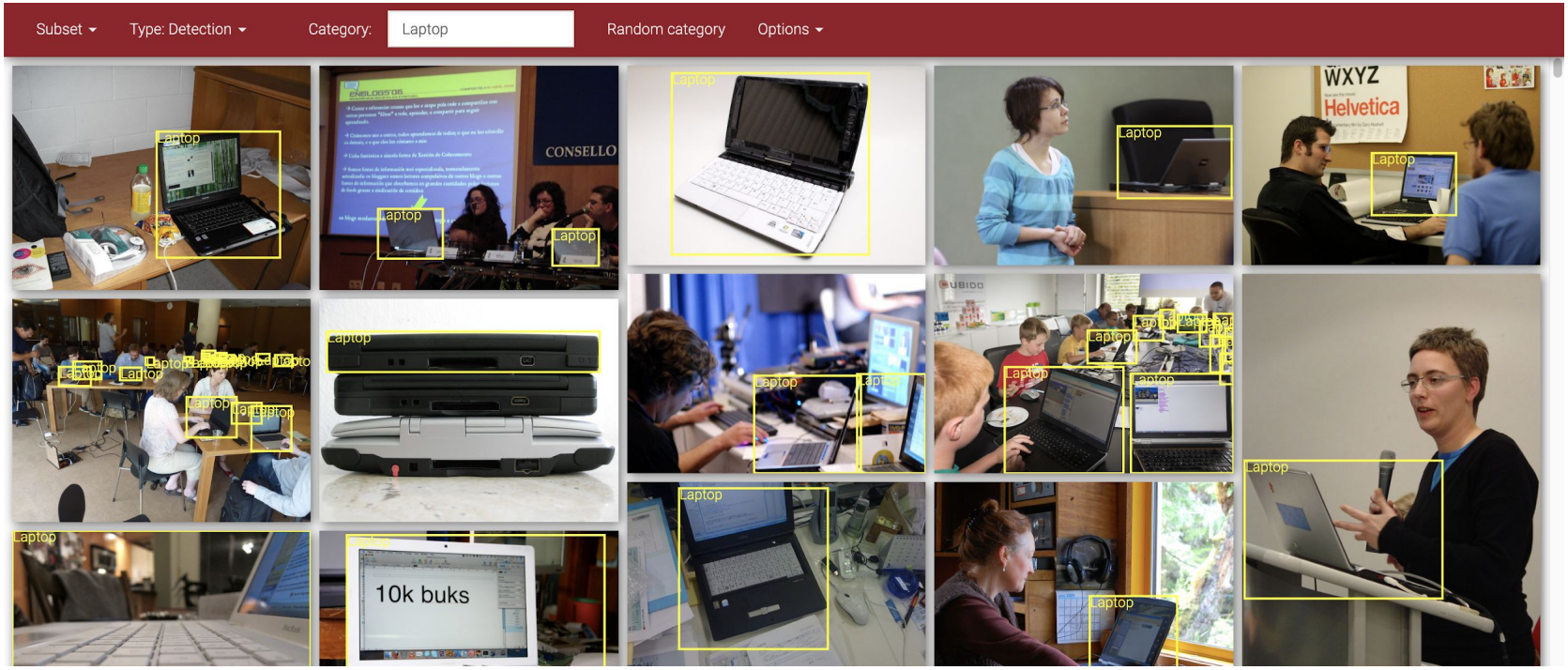
link:https://storage.googleapis.com/openimages/web/index.html

Downloader: OIDv4_ToolKit

Category: Car, Bicycle wheel, Traffic light, Person, Jeans, Laptop, Bus

Instance of each category: 1000

Dataset


Subset
Type: Detection
Category:
Laptop
Random category
Options
CONSELLO
WXYZ
Helvetica
10k buks

Data preprocessing

Needed files for darknet framework:

- annotation for each image
- train.txt
- test.txt
- data.data
- classes.names

Data preprocessing

Annotation.txt (each image has its own .txt):
class number
center x
center y
width
height

Data preprocessing

train.txt:

/full path/image1.jpg

/full path/image2.jpg

…..

test.txt:

/full path/image1.jpg

/full path/image2.jpg

…..

Data preprocessing

data.data:

classes=the number of classes in the dataset

train=full path to train.txt file

valid=full path to test.txt file

names=full path to classes.names file

backup=backup

classes.names:

Car

Bicycle wheel

.....

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Training YOLO in Darknet Framework

Setting up Configuration

Classes = 7

filters = (7 + 5) x 3 = 36

max_batches = 7 x 2000 = 14000

steps = 0.8 x max_batches / 0.9 x max_batches

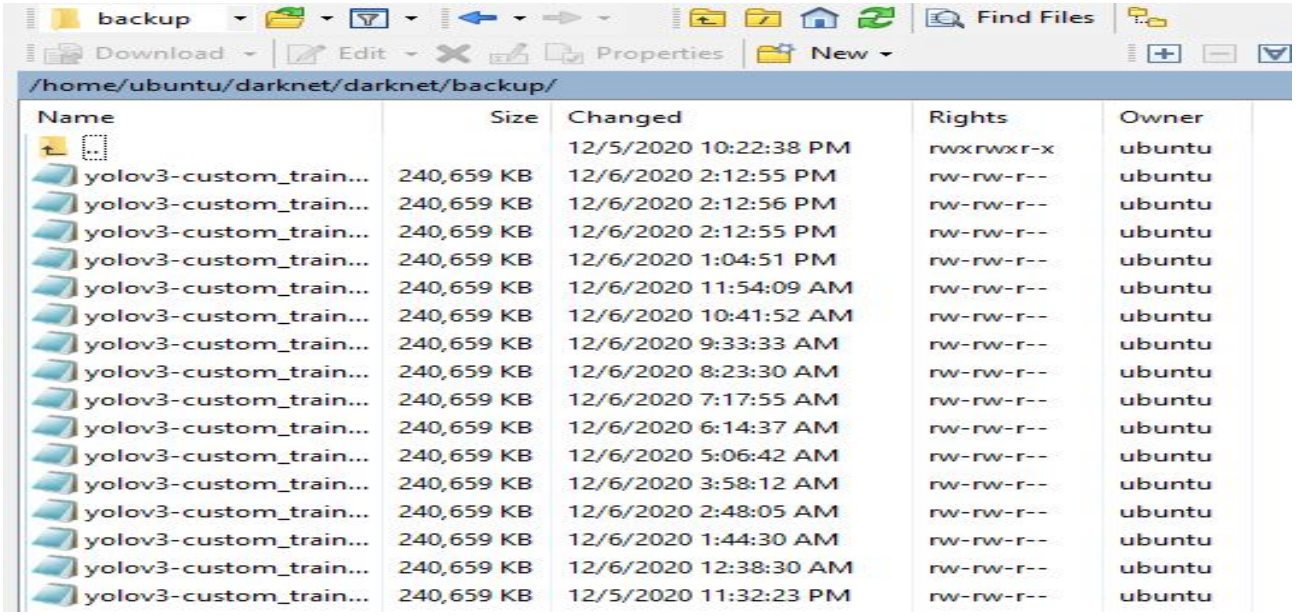
= 11200 / 12600

batch = 32

subdivisions = 8

minibatch = batch/subdivisions = 4

Result of Training Process



Name	Size	Changed	Rights	Owner
..		12/5/2020 10:22:38 PM	rwxrwxr-x	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 2:12:55 PM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 2:12:56 PM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 2:12:55 PM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 1:04:51 PM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 11:54:09 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 10:41:52 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 9:33:33 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 8:23:30 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 7:17:55 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 6:14:37 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 5:06:42 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 3:58:12 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 2:48:05 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 1:44:30 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/6/2020 12:38:30 AM	rw-rw-r--	ubuntu
yolov3-custom_train...	240,659 KB	12/5/2020 11:32:23 PM	rw-rw-r--	ubuntu

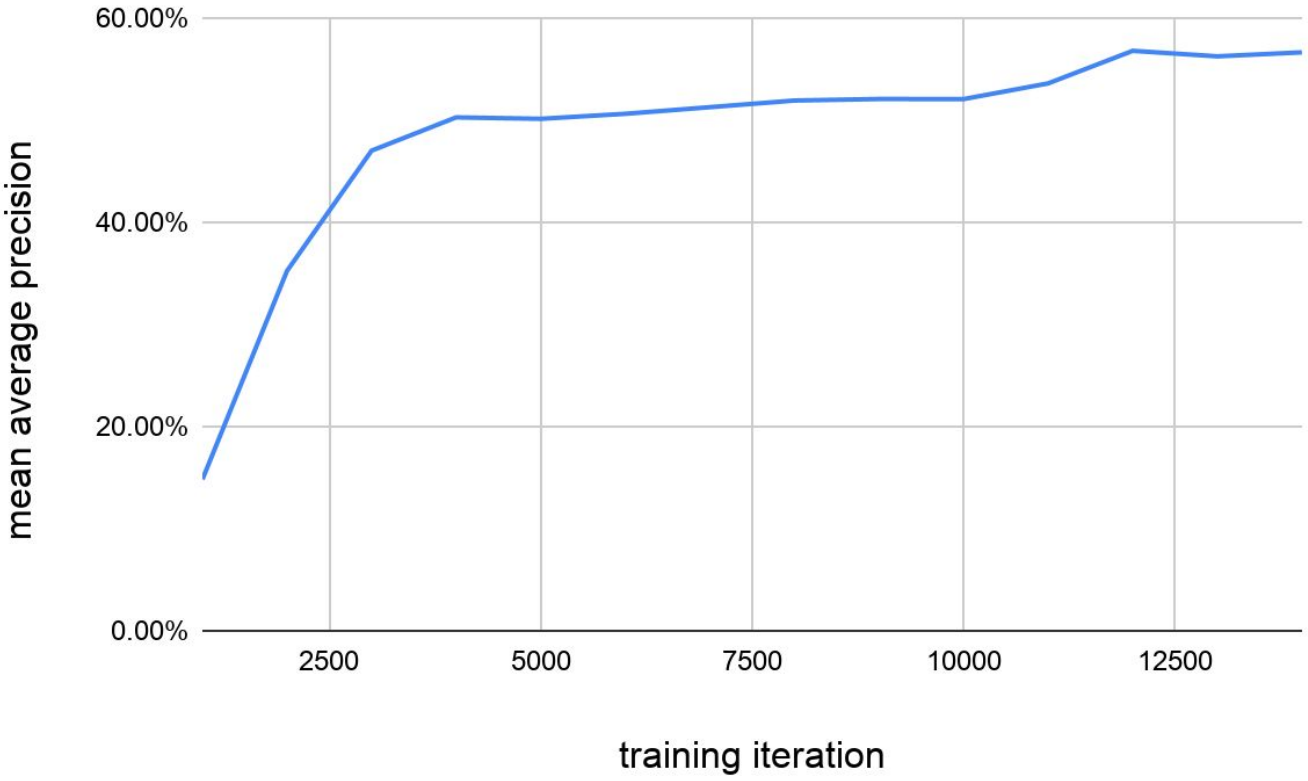
Mean Average Precision Example:

```
class_id = 0, name = Car, ap = 51.01%           (TP = 308, FP = 165)
class_id = 1, name = Bicycle wheel, ap = 62.46%         (TP = 304, FP = 68)
class_id = 2, name = Bus, ap = 79.25%           (TP = 161, FP = 35)
class_id = 3, name = Traffic light, ap = 44.14%         (TP = 329, FP = 222)
class_id = 4, name = Jeans, ap = 55.24%         (TP = 183, FP = 64)
class_id = 5, name = Laptop, ap = 80.80%        (TP = 155, FP = 28)
class_id = 6, name = Person, ap = 23.76%        (TP = 230, FP = 308)

 for conf_thresh = 0.25, precision = 0.65, recall = 0.45, F1-score = 0.53
 for conf_thresh = 0.25, TP = 1670, FP = 890, FN = 2047, average IoU = 50.15 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.566669, or 56.67 %
```

Performance of Different Training Weights



The best mAP is 56.82%, appears at iteration 12000.

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Apply The Best Model on Testing image and video

Loading Best Model

There are two different ways to do object detection:

- Using darknet framework's command line
  - './darknet detector test cfg/custom_data.data cfg/yolov3-custom_train.cfg backup/yolov3-custom_train_last.weights data/laptop-jean-test.jpg -out_filename data/result-laptop-jean-test.jpg -dont_show'
- Run process_image_video.py file
  - python3 process_image_video.py

Algorithm of process_image_video.py

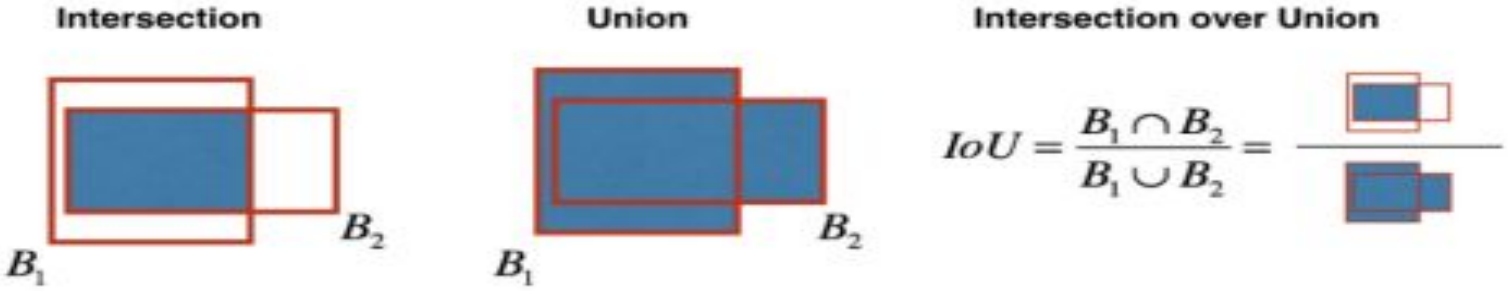
1. Read the input image and convert it into a blob object which is accepted by the YOLO framework.
2. Load the best trained weights and the network structure file into the YOLO framework.
3. Process blob image file into the model and generate output bounding boxes for each object inside the image.
4. Using Non-Maximun_Suppression technique on the output bounding boxes to drop duplicated boxes on the same object with lower confidence.

What the Results look like

```
out_scores, out_boxes, out_classes = predict(sess, "test.jpg")
```

```
Found 7 boxes for test.jpg
car 0.60 (925, 285) (1045, 374)
car 0.66 (706, 279) (786, 350)
bus 0.67 (5, 266) (220, 407)
car 0.70 (947, 324) (1280, 705)
car 0.74 (159, 303) (346, 440)
car 0.80 (761, 282) (942, 412)
car 0.89 (367, 300) (745, 648)
```

Non-Maximum-Suppression



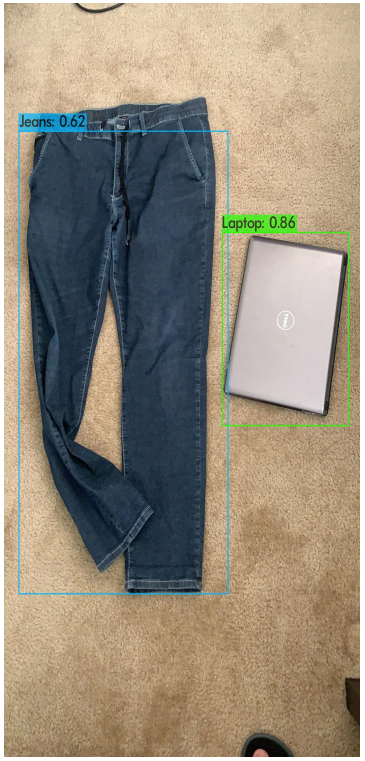
Results of Object Detected Images

Original Image

Model Detection

Results of Object Detected Images

Original Image



Model Detection

Results of Object Detected Images

Original Image

Model Detection

Results of Object Detected Videos

https://drive.google.com/file/d/1luAp5ggEG8hvPBlzG8yjvvBXh1PIXClW/view?usp=sharing

https://drive.google.com/file/d/1xc2RdESogHPag3VwwnbuNQs-RX--mZzF/view?usp=sharing

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Limitation and Conclusion

Limitations to YOLO

In spite of the high accuracy numbers and almost perfect bounding boxes, there are a few limitations to this object detection methods:

- Sometimes fails to detect overlapping objects, and objects that are partially visible in the frame.
- Detects the object class falsely if the features are a little blurred.
- It is very sensitive to model overfitting.

Conclusion and discussion

- The number of iteration of training process is not long enough to predict all categories we choose. (2000 iteration for each class)
- Our model is under-trained, compared to coco weight. Our dataset is not big enough to train a very accurate model
- Our model performs not well at detecting bus and person
- Our model has some difficult to detect object when there are multiple classes are overlapped. It will only detect one class.

References

Joseph Redmon, Santosh Divvala, Ross Girshick, ALi Farhadi “You Only Look Once: Unified, Real-Time Object Detection”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

Google open image: https://storage.googleapis.com/openimages/web/index.html