Training YOLOv3 to Detect Various Types of Vehicles

YC Feng

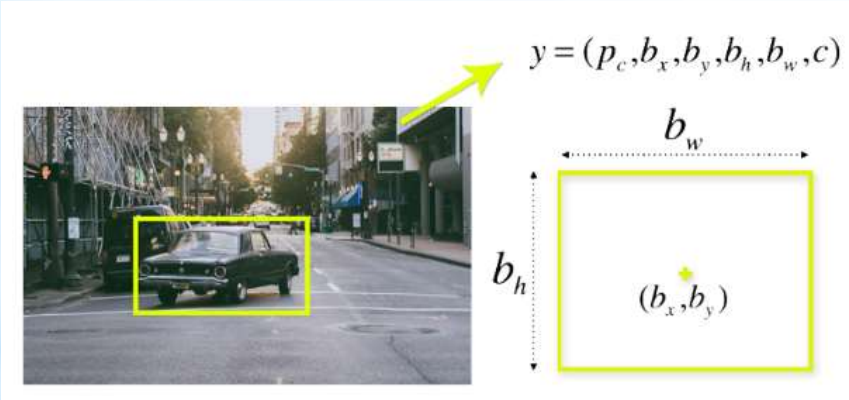
2020.7.4

Overview

1. Introduction – What is YOLO
2. Project Scope
3. Data Preparation
4. Set-up and Training
5. Results and Discussion
6. Future Works

What is YOLO?

- YOLO (“You Only Look Once”) is an effective real-time object recognition algorithm
- It predicts a class of an object and the bounding box specifying object location. Each bounding box can be described using four descriptors:



1. the probability that there is an object in the bounding box (pc)
2. center of a bounding box (bxby)
3. width (bw)
4. height (bh)
5. value c which corresponds to a class of an object

- Image is split into cells (typically using a 19×19 grid) and anchors are used to detecting multiple objects in a cell
- Boxes with low pc and high shared area (IoU) are removed (non-max suppression)

Computer Vision Problem Types

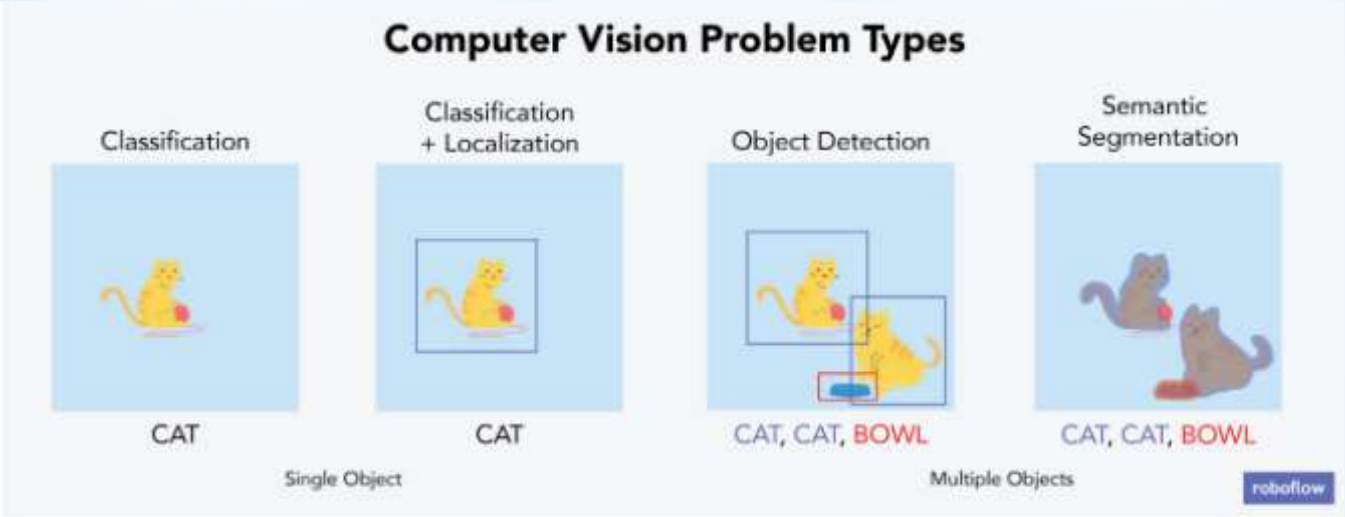


Image Processing Problems, adapted from Stanford's CS231N course

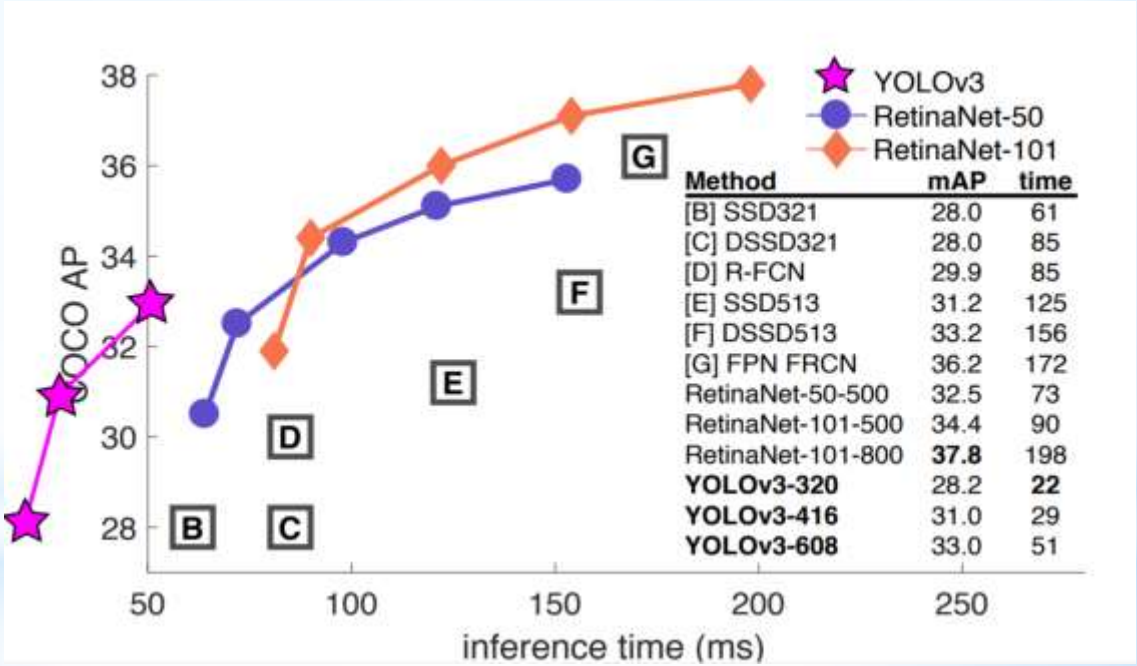
CNN

YOLO
Retina Net
SSD
DSSD
R-FCN
FPN FRCN

Mask RCNN
U-Net

Object Detection Methods Comparison

YOLO is the fastest!



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Comparison of Inference time between YOLOv3 with other systems on COCO dataset
Source

Project Scope

- We train a YOLOv3 model to detect 8 types of vehicles:
  1. Ambulance
  2. Fire Truck
  3. Fedex
  4. Bus
  5. Police Car
  6. UPS
  7. USPS
  8. Others (sedan, train, trucks, etc.)

Data Preparation

- More than 600 pictures (75 – 80 for each group) are used for training.
- Either manually download individual images from web or use console.js to download image from https://images.google.com .
- Use image annotation/labeling tool https://github.com/tzutalin/labelImg to specify bounding box(es) to cover areas of classified objects of each image. A .txt file is generated, which defines object class, center positions, width and height for each object.
    - **For Window 10 PC, follow steps as:**

    - **For Mac OS, setup a git bash then do below**

      git clone https://github.com/tzutalin/labelImg.git

Special thanks: Gloria Sun and Peng Lin for annotation. I also borrowed 200 pictures from Patrick Li.

Set-up and Training

1. Colab (with GPU) is used, see the notebook for more details. → 

2. The following files are also needed, sample files are attached.
   - yolov3.cfg / yolov3_test.vfg → 

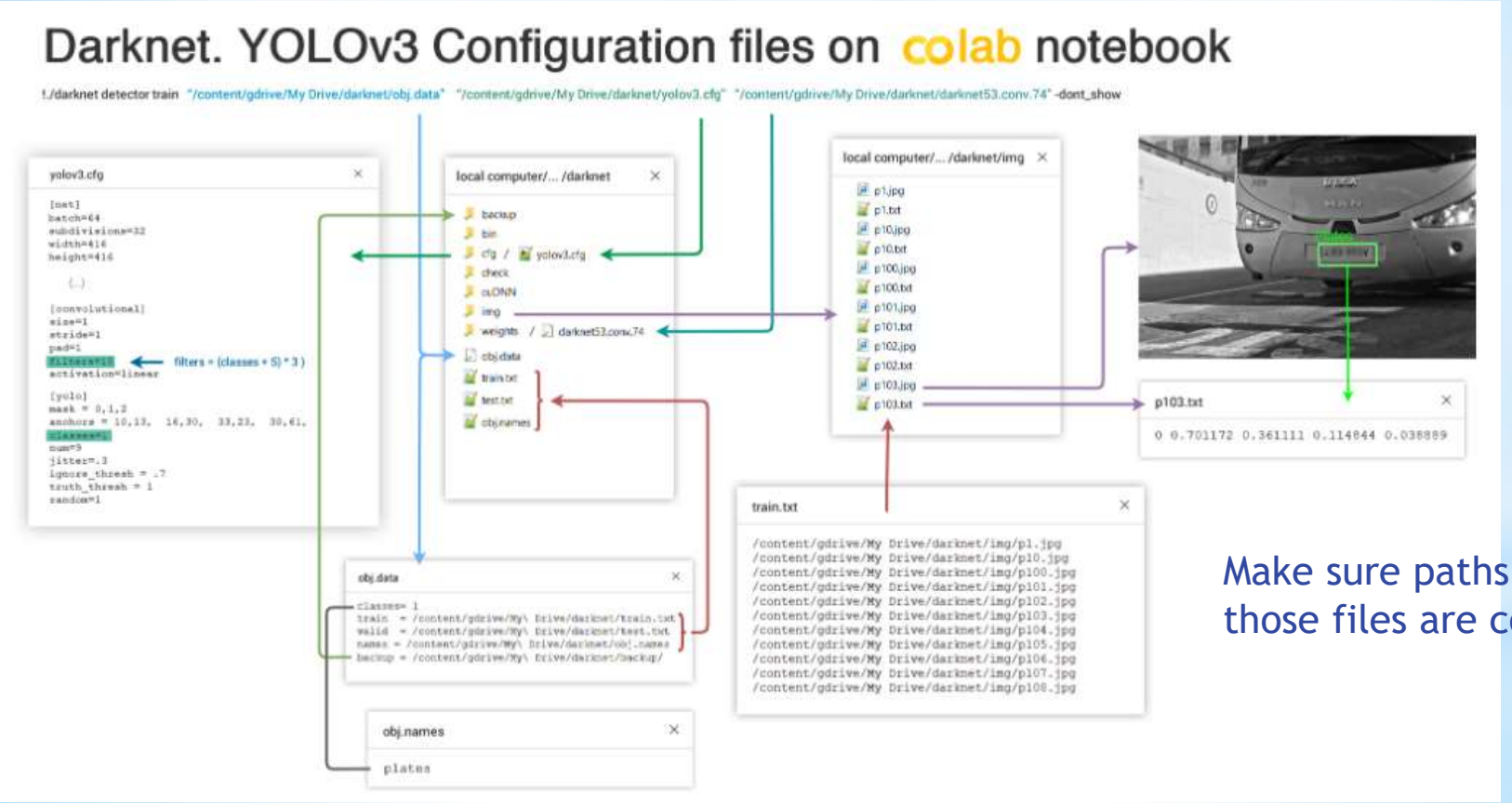
   - train.txt / test.txt → 

   - obj.data → 

   - obj.name → 

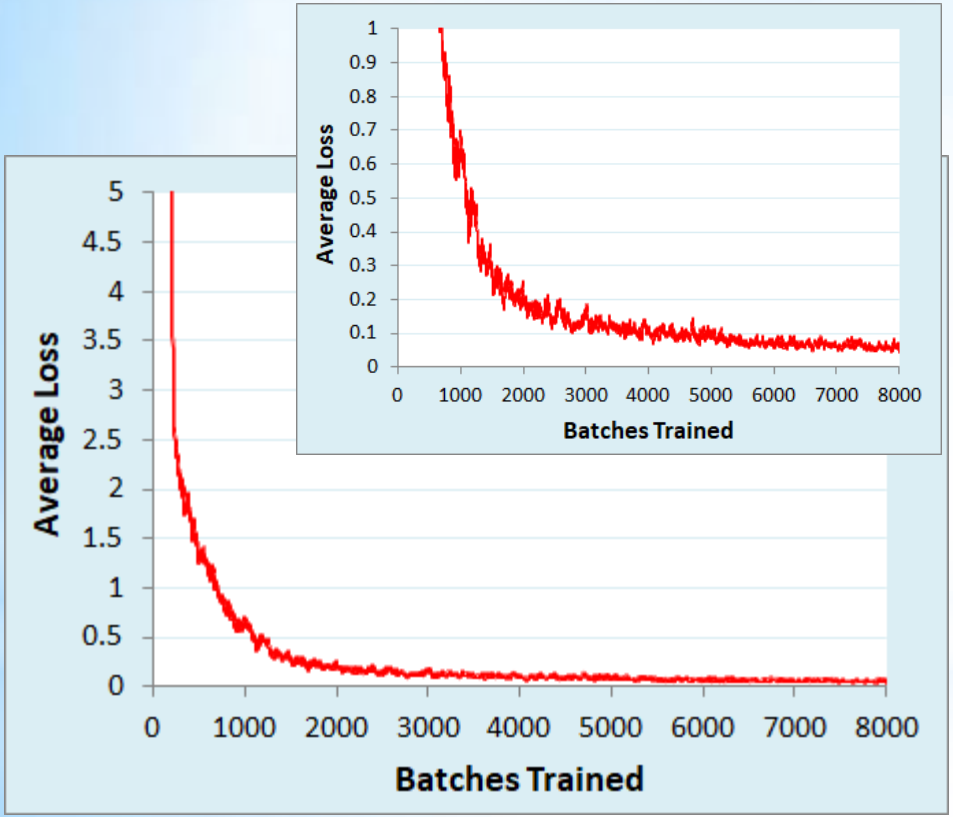
   - darknet53.conv.74 (downloaded)
3. Trained 9000 batches.

Set-up and Training - Continue



Make sure paths to those files are correct.

Results and Discussion – Average Loss



- Average loss reduces quickly with increasing numbers of batches trained.
- Over 4000 batches, the average loss continue improve, but at much slower rate.

Results and Discussion – Pictures

Successful

Fail to detect

- Detections on “ideal” conditions are successful.
- However, detection in some “challenge” conditions failed.

Results and Discussion – Video

- We also use a video to test our model →
- Similar to the pictures, in clear conditions detection are successful.
- Detection in “challenge” conditions (at night) is not so successful.

Env_detection3.mp4

Detected

Failed to detect

Summary and Ideas for Further Work

- A YOLO model is trained to detect 8 types of vehicles. It works reasonably well in idea conditions. However, it is not very successful in some challenge conditions. A notebook is attached at page 8.

- Here are some ideas for further work:
  - Train the model with more samples, especially more sample with different challenge conditions, with siren lights, etc.
  - Explore splitting the “Others” group – putting ‘sedan’ and ‘train’ in the same group might make the machine learning harder
  - Use augmentation to further increase sample size and varieties.
  - Reduce data size (resolution) for higher speed
  - Out of Box: integrate with sound to detect emergent vehicles
  - …

References

Inference / Demo

- https://towardsdatascience.com/yolov3-pytorch-on-google-colab-c4a79eeecdea
- https://medium.com/@artinte7/real-time-object-detection-using-yolo-upon-google-colab-in-5-minutes-fd65a4903df5

Training / Implement

- https://github.com/ultralytics/yolov3
- https://github.com/BlcaKHat/yolov3-Helmet-Detection
- https://towardsdatascience.com/training-a-yolov3-object-detection-model-with-a-custom-dataset-4981fa480af0
- https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/
- https://medium.com/@quangnhatnguyenle/how-to-train-yolov3-on-google-colab-to-detect-custom-objects-e-g-gun-detection-d3a1ee43eda1
- https://www.learnopencv.com/training-yolov3-deep-learning-based-custom-object-detector/
- https://github.com/eriklindernoren/PyTorch-YOLOv3
- http://blog.ibanyez.info/blogs/coding/20190410-run-a-google-colab-notebook-to-train-yolov3-using-darknet-in/
- https://blog.francium.tech/custom-object-training-and-detection-with-yolov3-darknet-and-opencv-41542f2ff44e
- https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

Darknet

- https://pjreddie.com/media/files/darknet53.conv.74
- https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects

Further Improvement

- https://github.com/srp-31/Data-Augmentation-for-Object-Detection-YOLO-
- https://towardsdatascience.com/data-augmentation-in-yolov4-c16bd22b2617
- https://arxiv.org/pdf/1804.02767.pdf;

Algorithm

- https://appsilon.com/object-detection-yolo-algorithm/
- https://www.youtube.com/watch?v=9s_FpMpdYW8

Thank you!