Training detection models on custom data

CV&DL Homework 2

Artem Bakhanov
Innopolis University
Innopolis, Russia
a.bahanov@innopolis.university

***Abstract*—This document describes the process of training different detection and instance segmentation deep learning models on customly gathered dataset.**

***Index Terms*—dataset, yolo, yolov4, yolov5, mask-rcnn, detection, segmentation**

I. Introduction

This paper describes the process of collecting and annotating custom dataset for object detection and instance segmentation. Then I describe the process of training several deep learning models (YoloV4 [1], YoloV5 [2], Mask-RCNN [3]) and compare them with different metrics. GitHub repository: https://github.com/artembakhanov/cvdl-object-detection.

II. Dataset

The dataset consists of 168 images containing cameras and smoke detectors. The dataset was split into 3 parts: 144 images in training set, 16 in validation and 8 in test sets. An example is provided on Fig. 1.

Fig. 1: Image example containing one camera instance (on the bottom) and one smoke detector instance (on the top).

A. Data Acquisition

The data was gathered using built-in camera on realme C25s phone inside the main building of Innopolis University. Most of the images were taken with flash since most of the places had relatively bad light conditions. Then all the photos were downscaled from 4000×3000pxs to the width of 1024pxs. However, the data were collected twice and the first time it was done the pictures did not have enough quality. Also the objects on those images were very small and almost impossible to recognize even for a human being.

B. Object detection labeling process

Before passing the acquired images to the models I needed to label all the images manually. For this purposes, a web application for data annotation Supervisely [4] was used. After labeling objects and instances the result dataset was put into Roboflow software [5] in order to convert Supervisely format to YoloV4 and YoloV5 specific formats. Additionally, Roboflow provides an instrument for an easy in-code dataset downloading, which was used for training too. Instance segmentation dataset was not converted to any specific format, a custom dataset in PyTorch [6] was created in order to read it. Examples of labeled data are provided on Fig. 2 and Fig. 3.

Fig. 2: Training example for object detection models. Blue bounding box: smoke detector; red bounding box: camera.

III. Training process

All of the models were trained on Yandex.DataSphere machines with GPU NVidia Tesla V100 accelerators. All IPython Notebooks with training code are available on my repository https://github.com/artembakhanov/cvdl-object-detection.

Fig. 3: Training example for instance segmentation models. Blue region: smoke detector; red region: camera.

A. YoloV4

Scaled YoloV4 model was trained for 100 epochs with batch size 16 on the gathered dataset rescaled to 416x416. The training pipeline was taken from the Roboflow tutorial [7]. The results are provided on Fig. I

TABLE I: YoloV4 Performance

Metric	Score
Precision	0.3895
Recall	1.0
mAP@0.5	0.817
mAP@0.5:0.95	0.463

B. YoloV5

Small YoloV5 model was trained for 100 epochs with batch size 16 and provided pretrained weights on the gathered dataset rescaled to 416x416. The training pipeline was taken from the Roboflow tutorial [8]. The results are provided on Fig. II

TABLE II: YoloV5 Performance

Metric	Score
Precision	1.0
Recall	1.0
mAP@0.5	0.995
mAP@0.5:0.95	0.775

C. Mask-RCNN

Mask-RCNN (with YoloV4 backbone) model was trained for 10 epochs with batch size 2 and provided pretrained weights from PyTorch on the gathered dataset rescaled to 416x416. The training pipeline was taken from the torchvision tutorial [9]. The results are provided on Fig. III.

TABLE III: Mask-RCNN Performance

Metric	Score
mAP@0.5 (bounding boxes)	0.972
mAP@0.5:0.95 (bounding boxes)	0.847
mAP@0.5 (segmentation)	0.972
mAP@0.5:0.95 (segmentation)	0.893

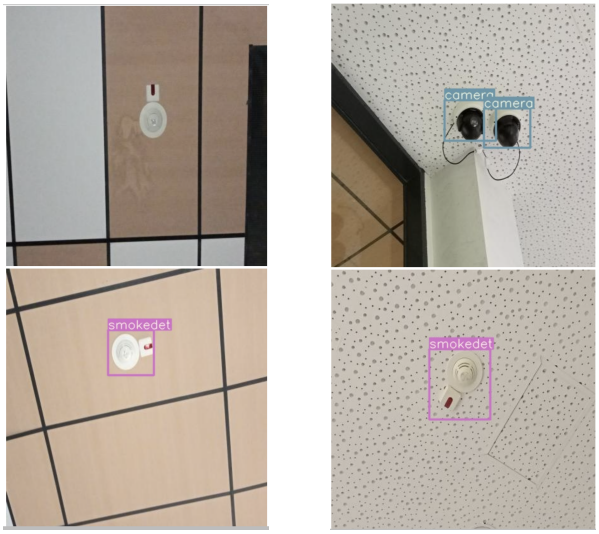

Fig. 4: YoloV4 inference example

IV. Results

Despite the fact that the size of the dataset was small all the models showed relatively good results. The worst model in terms of mAP metric was YoloV4. There were an example of image in the test set for which the model was not able to detect any of the object. The quality of bounding boxes are also not so good. The segmentation model was good even in predicting bounding boxes, the results are comparable with YoloV5s results. Those two models showed very good results even on small dataset. Inference examples are shown on Fig. 4, Fig. 5 and Fig. 6.

V. Discussion

All of the model performed well on the acquired dataset. The problem is that the quality of the dataset may be very high comparing to real-life examples. Also the fact that all of the images were taken on one device is not promising, however it is solvable with adding perturbations and augmentations to the images. Overall, in general I can say that I achieved the expected results.

VI. Conclusion

I have trained three different models for object detection and instance segmentation on a relatively small dataset. All of three models showed good results and all the desired gools were achieved.

Fig. 5: YoloV5 inference example

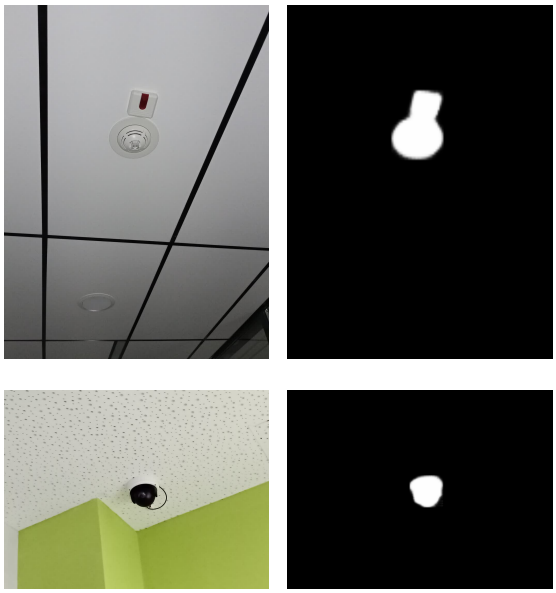

Fig. 6: Mask-RCNN inference example

ACKNOWLEDGMENT

I am very thankful to my professor of Computer Vision and Deep Learning at Innopolis University, Imad Eddine Ibrahim BEKKOUCH. He was very supportive during the whole course.

REFERENCES

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[2] G. Jocher, A. Stoken, A. Chaurasia, J. Borovec, NanoCode012, TaoXie, Y. Kwon, K. Michael, L. Changyu, J. Fang, A. V, Laughing, tkianai, yxNONG, P. Skalski, A. Hogan, J. Nadar, imyhxy, L. Mammana, AlexWang1900, C. Fati, D. Montes, J. Hajek, L. Diaconu, M. T. Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106, "ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support," Oct. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5563715

[3] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[4] "Supervisely." [Online]. Available: https://app.supervise.ly/

[5] "Roboflow." [Online]. Available: https://roboflow.com/

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[7] "Roboflow-custom-scaled-yolov4." [Online]. Available: https:\/\/colab.research.google.com\/drive\/1LDmg0JRiC2N7_tx8wQoBzTB0j\UZhywQr

[8] [Online]. Available: https:\/\/colab.research.google.com\/drive\/1gDZ2xcTOg\R39tGGs\protect\discretionary{\char\hyphenchar\font}{}{}EZ6i3RTs16wmzZQ

[9] "torchvision_finetuning_instance_segmentation." [Online]. Available: https://colab.research.google.com/github/pytorch/tutorials/blob/gh\protect\discretionary{\char\hyphenchar\font}{}{}pages\/_downloads\/torchvision_finetuning_instance_segmentation.ipynb