[F21] Computer Vision and Deep Learning

Few-Shot instance segmentation or YOLOv4 tiny, YOLOv5s and Mask R-CNN trained on tiny poorly annotated dataset

Yaroslav Shumichenko

December 4, 2021

Keywords
Object detection, Semantic Segmentation, YOLOv4, YOLOv5, Mask R-CNN, misery, frustration

1 Introduction

Data is very important for machine learning. For this assignment, we have manually collected and annotated a dataset of the most common object that can be found on the route from Technopark to dormitories in Innopolis - cars and trees. Images were uploaded to Supervisely platform and annotated for instance segmentation task, that is, each object of interest on the image is surrounded by a polygon. It allows to export annotations to different formats - object detection for YOLO and instance segmentation for Mask R-CNN. We used Roboflow to convert the data to YOLO object detection format and train YOLOv4 tiny and YOLOv5s models. For Mask R-CNN, we used tools provided by Supervisely for dataset preparation and model training. As expected, we obtained poor (and not even representative due to amount of data) results.

2 Materials & Methods

As we outlined previously, we heavily utilized Suprvisely and Roboflow platforms. Supervisely provides a convenient annotation tool, as well as its own model zoo and model training platform, which allows to install a Supervisely agent on your PC and train models inside nvidia docker. Roboflow provides a dataset management tool and model zoo as well - a dataset can be uploaded in different formats and converted automatically. Moreover, it can perform preprocessing steps - such as image resizing and various augmentations to increase dataset size. Roboflow's model allows you to fetch the dataset from google colab notebooks with all neccessary code to train models. For YOLOv4 tiny and YOLOv5s models, we used Roboflow dataset tool to prepare and augment the dataset and used Roboflow model zoo to train them. For Mask R-CNN, a Supervisely model zoo was used to train the model on local PC inside a Supervisely agent.

3 Results

Images for dataset were collected in Innopolis at the evening using ancient Nokia 7.1 camera, which captures low quality 4032x3024 images. The example of taken image can be seen on Figure 1. After filtering really bad images, a total of 42 images with 123 'Car' objects and 55 'Tree' objects were left. For a bit of diversity, we also included some images from Kazan to the dataset.

Images were uploaded to Supervisely annotation tool and labeled for an instance segmentation task. An example of labeled image is on Figure 2.

To prepare data for YOLO models, the dataset from Supervisely was uploaded to Roboflow. Resize to 416x416 with preserved aspect ratio was used for preprocessing, random horizontal flip, random crop and random brightness adjustment were used to produce 3 images per 1 in training set. 10 images were chosen for validation set. Due to the lack of data, we did not create a separate testing set and used validation set for testing. After that, the dataset was exported in YOLO Darknet format for YOLOv4 tiny and in YOLOv5 Pytorch

Figure 1: Not only the images themselves are bad, but compression from such high resolution makes them even more terrible

Figure 2: Due to author's laziness, some occluded objects were not labeled. Also the quality of masks for class 'Tree' leaves much to be desired

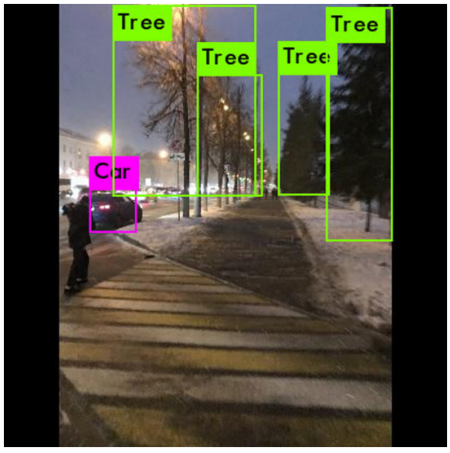

Figure 3: Inference example of YOLOv4 model

Figure 4: Inference example of Mask R-CNN

format for YOLOv5s.

To train YOLO object detection models, we utilized Google Colab notebooks provided in Roboflow model zoo. A YOLOv4 tiny model trained on 96 images achieved 48.51% mAP@0.5 on validation set after 2000 epochs of training (the model converged much earlier, this is the result of the best tracked checkpoint). A YOLOv5s model trained on 96 images achieved 59.17% mAP@0.5 on validation set after 2000 epochs of training (again converged on 300th epoch). The tensorboard graphs of YOLOv5s can are presented on Figures 5 and 6.

The example detection can bee seen on Figure 3.

The Mask R-CNN model was trained using Supervisely platform. The dataset was split into train-val using Supervisely DTL and trained using Supervisely agent installed on 2070SUPER GPU PC. The model was trained for 100 epochs, and the best one (30 epochs) was chosen for inference. Training graph can is presented on Figure 7. Example inference on image from validation set can be viewed on Figure 4.

4 Discussion

Actually, the model results were not so bad as according to metrics report. The sample size of validation set - 10 images - is way too small to adequately assess model's performance. Also, poor validation metrics can be described with insufficient effort put into labeling - some occluded and small object were not labeled, despite having features of other labeled objects of the same class. Models were able to find such objects, but due to the absence of labels they were counted as false positives, decreasing validation metrics. Thus the choice of best model according to validation metrics is not optimal, as model with worse metrics may actually perform better.

Conclusions

- Dataset collection is very important step. Insufficient data and incorrect labeling may result in incorrect interpretation of experiment results.
- Supervisely and Roboflow platforms provide amazing free services for fast prototyping and quality estimate of neural networks for computer vision.

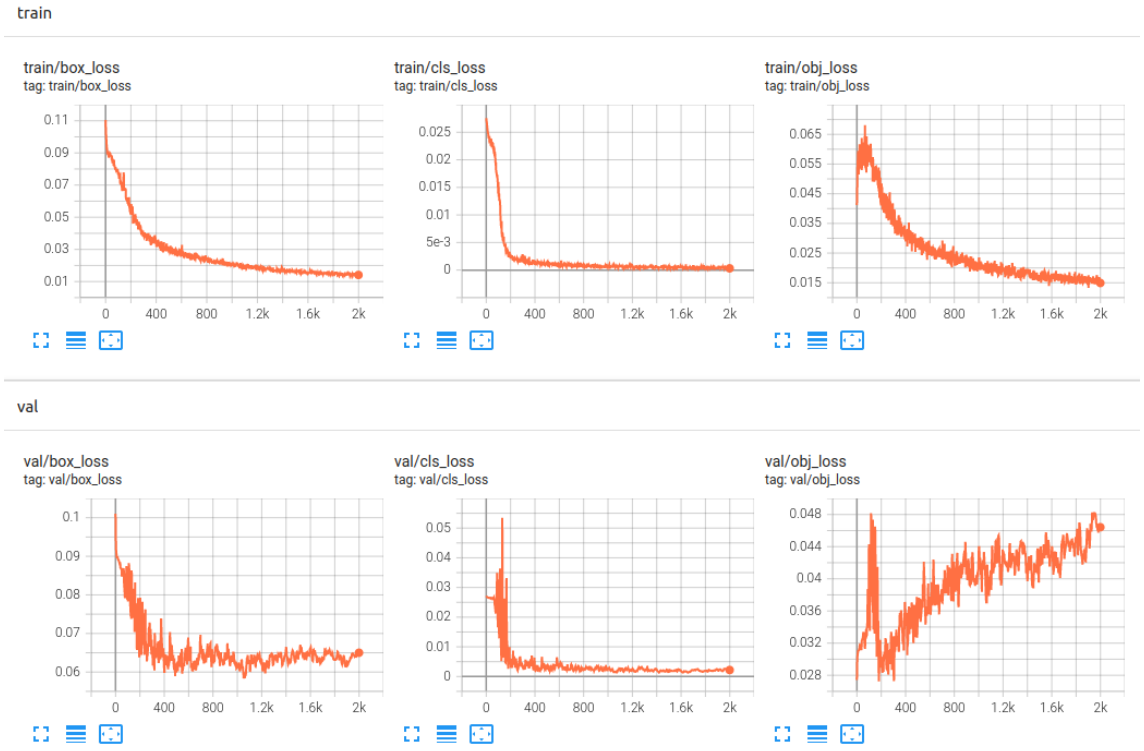

Figure 5: Loss graphs of YOLOv5s training process

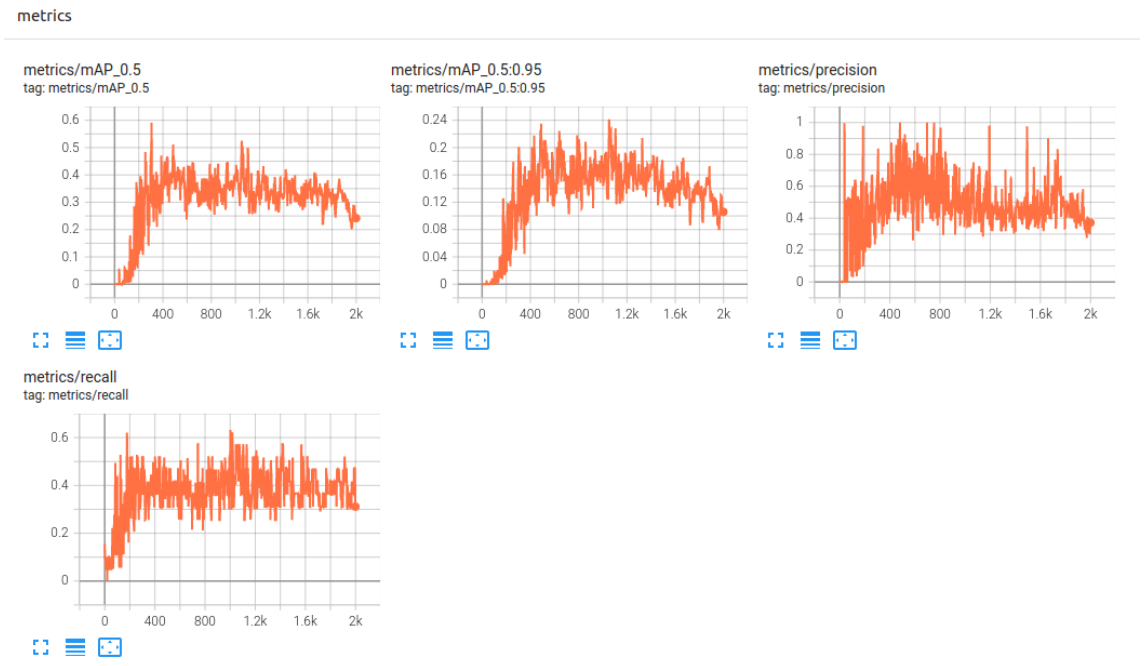

Figure 6: Metrics graphs of YOLOv5s training process

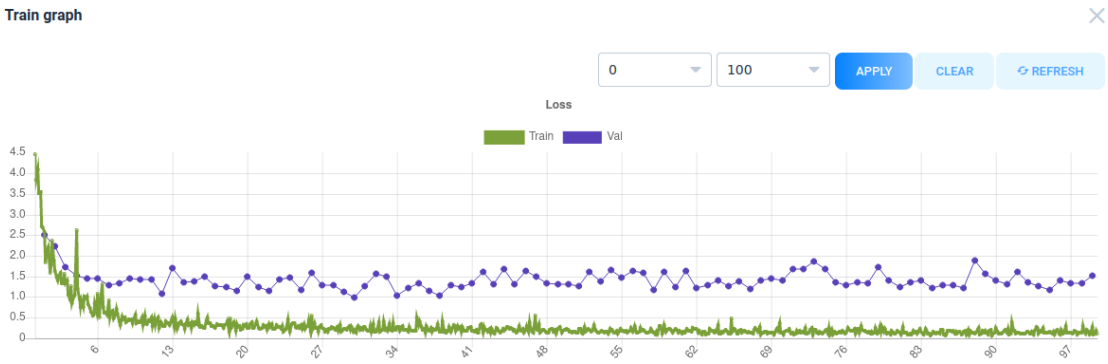

Figure 7: Loss graph of Mask R-CNN