REPORT OUT: PARKING LOT OCCUPANCY DETECTION USING YOLOv5

30/11/2021

Giuseppe Marino

Target Audience: Parking lot of a commercial centre in London

CONTENTS

- Review of Questions to Answer/ Hypotheses / Approach
- Discuss technical challenges
- Dataset preparation and augmentation
- Initial Findings
- Deeper Analysis
- Implementation
- Hypothesis Results

SECTION 1: QUESTIONS TO ANSWER

Parking lot occupancy detection deployed in a commercial centre :

- Generalized detection protocol working for occlusions and new parking lots.
- Avoid high-computing power detrimental over the training phase
- Using cost-effective and versatile sensors
- Decentralized strategy for decision making, ideal for closed parking lots
- Real time detection

SECTION 2 : INITIAL HYPOTHESIS

- Thanks to the use of a complete dataset and a deep CNN, the solution is expected to be robust to occlusions and variation of light conditions. Moreover, it is expected to generalize well on different parking lots
- If the training of CNN is carried out beforehand and only the inference phase is delegated to the device, the in-situ detection will need low computational resources.
- We expect that using smart cameras is cheaper than ground sensors as it can monitor more than one parking spot at the same time and it has lower installation costs. Complementary activities, such as face/people recognition, or tracking and logging of people and moving vehicles can be potentially added to the device.
- If we delegate the detection inference to smart devices we won't need the central server to upload the new taken pictures and carry out further elaboration, so this detection solution will be easily scalable.

SECTION 3: APPROACH

- **Training model.** For smart cameras deployments we have chosen a relatively low computing power object detection model such as YOLOv5s/m. It is based on the YOLOv5 repository by Ultralytics, the Pklot dataset, the data management tools by Roboflow and the blog post on how to train YOLOv5.
- **Implementation** Cameras are installed in the parking lot of a commercial center in strategic positions as to take into account different weather conditions and occlusions, following the approach on: Amato, G., et al. (2017). Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, *72*, 327-334.

SECTION 3: APPROACH – TRAINING DATASET

- **Yolov5s will be pretrained with the COCO dataset.** Its images contain enough small features typical of shallow layers.
- For the training of more complex features in deeper layers**, YOLOv5s will be trained with the Pklot dataset*: Images per class.** ≥1.5k images per class. **Instances per class.** ≥10k instances (labeled objects) per class total. **Image variety.** Must be representative of deployed environment. For real-world use cases we recommend images from different times of day, different seasons, different weather, different lighting, different angles, different sources (scraped online, collected locally, different cameras) etc. **Label consistency.** All instances of all classes in all images must be labelled. Partial labelling will not work. **Label accuracy.** Labels must closely enclose each object. No objects should be missing a label. **Background images.** Background images are images with no objects that are added to a dataset to reduce False Positives (FP).

* Almeida, P.R.D., et al. PKLot—A robust dataset for parking lot classification. Expert Syst. Appl. **2015**, 42, 4937–4949.

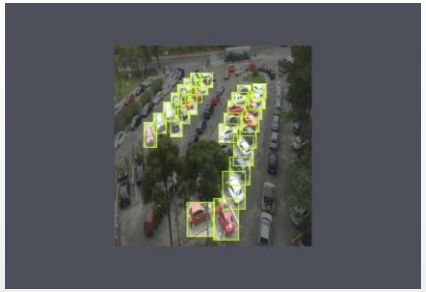
SECTION 3: APPROACH – YOLOv5 TRAINING

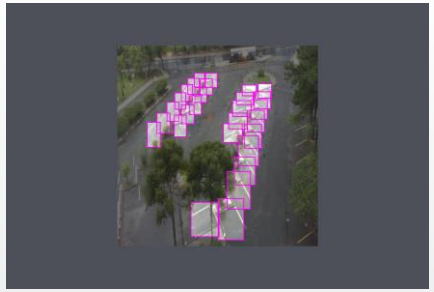
- **Epochs.** Start with 300 epochs. If this overfits early then you can reduce epochs. If overfitting does not occur after 300 epochs, train longer, i.e. 600, 1200 etc epochs.
- **Image size.** COCO trains at native resolution of --img 640, though due to the high amount of small objects in the dataset it can benefit from training at higher resolutions such as --img 1280. If there are many small objects then custom datasets will benefit from training at native or higher resolution. Best inference results are obtained at the same --img as the training was run at, i.e. if you train at --img 1280 you should also test and detect at --img 1280.
- **Batch size.** Use the largest --batch-size that your hardware allows for. Small batch sizes produce poor batchnorm statistics and should be avoided.
- **Hyperparameters.** Default hyperparameters are in hyp.scratch.yaml. We recommend you train with default hyperparameters first before thinking of modifying any. In general, increasing augmentation hyperparameters will reduce and delay overfitting, allowing for longer trainings and higher final mAP. Reduction in loss component gain hyperparameters like hyp['obj'] will help reduce overfitting in those specific loss components. For an automated method of optimizing these hyperparameters, see our Hyperparameter Evolution Tutorial. SGD is the default optimization algorithm.
- **How to find a good model** in two stages: first get a model large enough that it can overfit (i.e. focus on training loss) and then regularize it appropriately (give up some training loss to improve the validation loss).

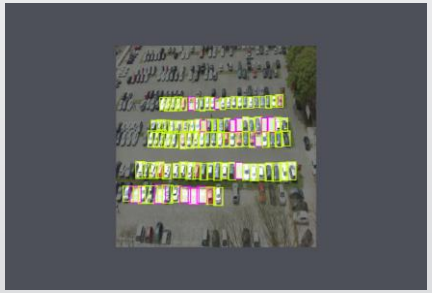
TECHNICAL CHALLENGES

- For reducing computation time, the model available online contains only 160 images from the Pklot dataset in different weather conditions and different parking lots. Augmentated data are also generated during training.
- In order to generalize well, validation, test and 15% of training examples have been chosen from a different parking lot than the 85% of training examples.
- The size of images has been changed in Roborflow to match the YOLOv5 input format of 416
- The number of epochs has been chosen to be 300 as at this point, all losses (classification, object, and box) have reached a minimum plateau.

LABELLED AND AUGMENTED DATA IN TRAINING SET:
two parking lots at different weather conditions

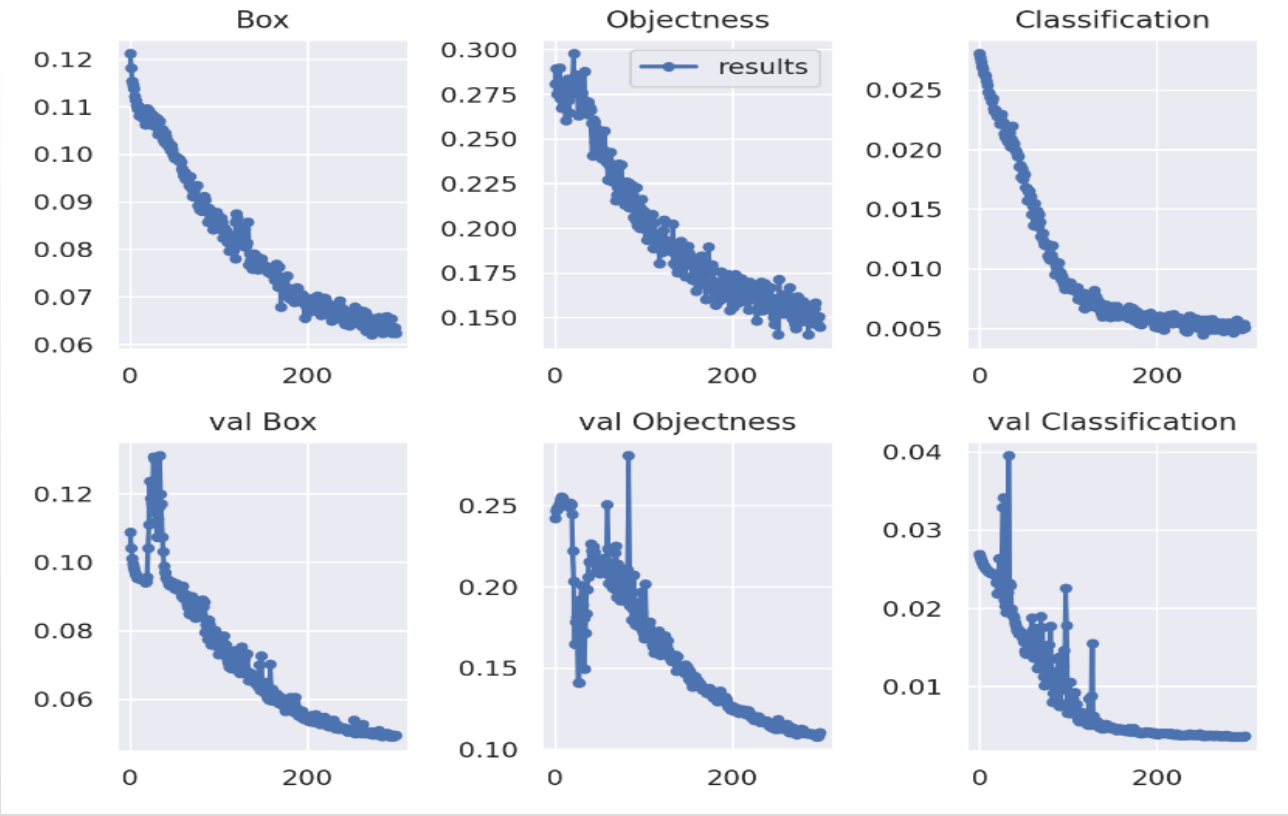






LABELLED DATA IN VALIDATION AND TEST SETS:
a third parking lot at different weather conditions

INITIAL FINDINGS : overfitting and regularization OK

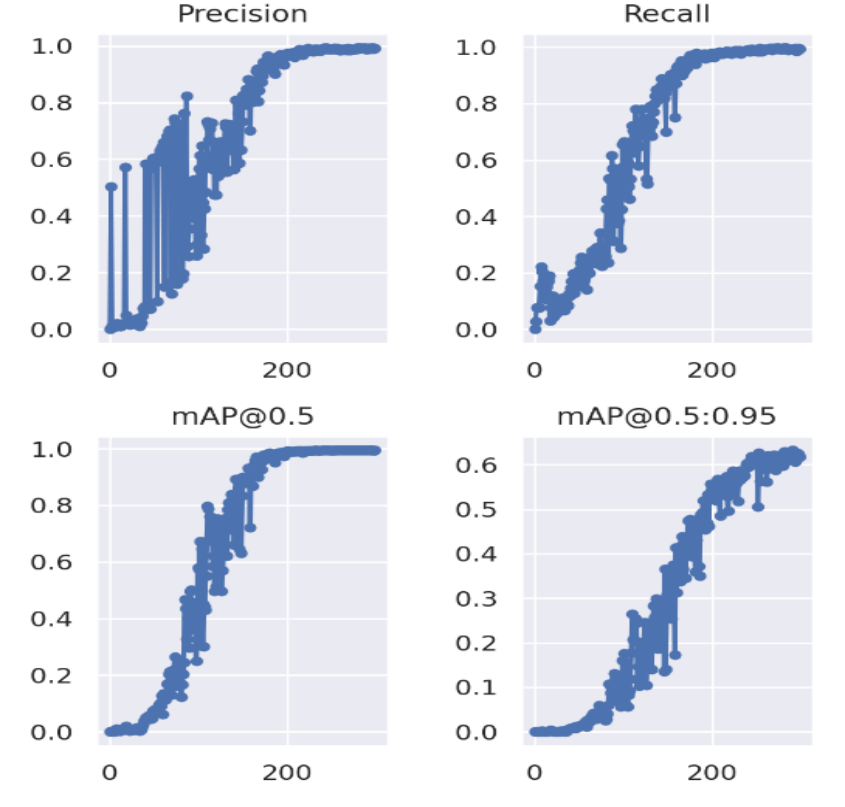


Training losses (first row) for all three parameters: box, object and classification reach a minimum plateau which is a sign of overfitting.

As validation losses (second row) are smaller than training losses, regularization is working fine.

The above means that we have chosen good hyperparameters for our model.

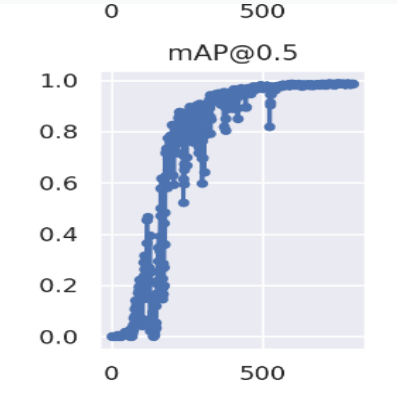
INITIAL FINDINGS : accuracy OK



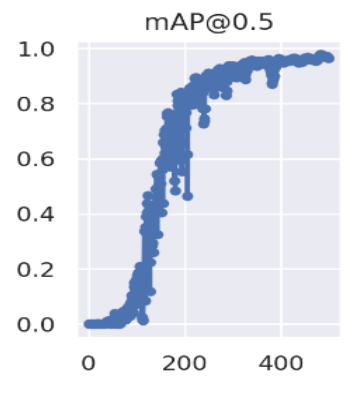
Precision, Recall, Mean Avearge Precision and Mean Avearge Precision .5-.95 reach a maximum plateau which is another sign that regularization is working fine and that we have chosen good hyperparameters for our model.

DEEPER ANALYSIS : pretraining

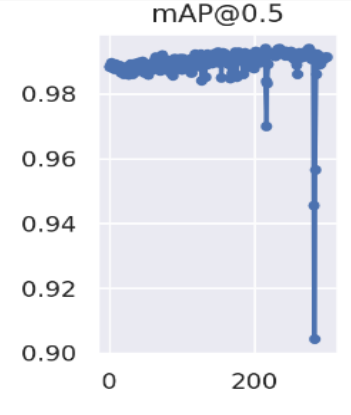
Random Weights



COCO Weights

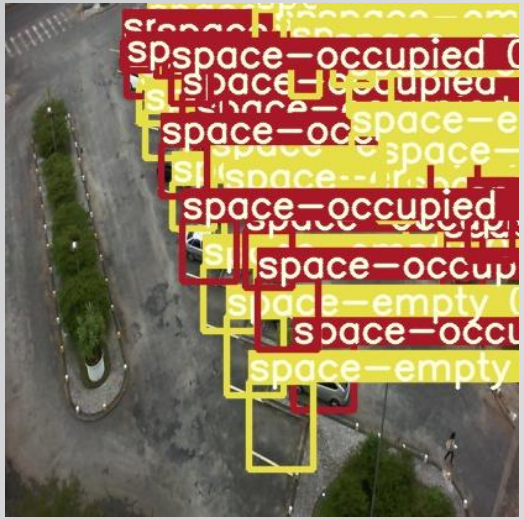


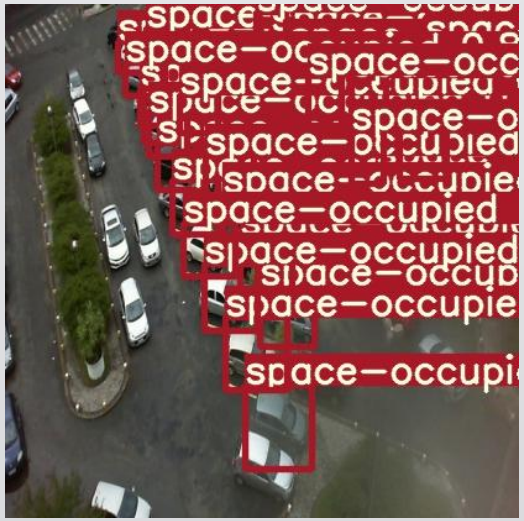
Custom Pklot Weights

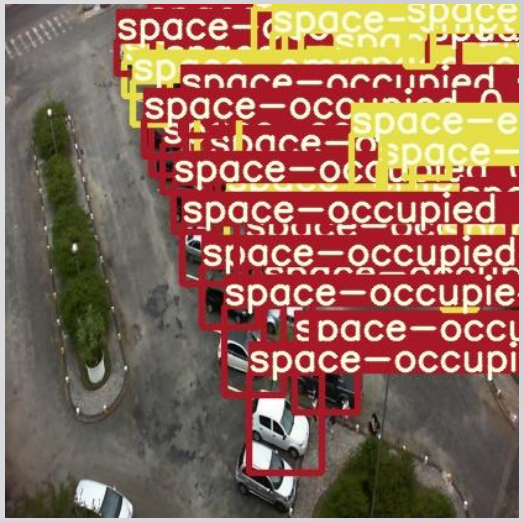


In this slide we have analysed the importance of pretraining: after 300 epochs the mAP is respectively 0.80 0.898 and 0.993 for the three different weights. This means that COCO is better than random pretraining as it contains shared small features with the custom data. Further improvement is achieved if we use weights issued from a model pretrained with COCO weights and trained with custom data as in this case more complex features are shared.

DEEPER ANALYSIS: Run inference with trained weights from the COCO pretrained model







As can be seen in figure, our approach generalize well to a new parking lot and deals well with common challenging visual classification problems such as obstacles (other vehicles parked closely or transiting).

IMPLEMENTATION – Decentralized Inference

- We have chosen a decentralised test inference carried out by smart cameras. Such cameras capture the image of a portion of the parking lot, anchor the parking spaces and, for each of them determine the occupancy status by using a CNN trained off-line.
- Smart cameras are equipped with a module Raspberry Pi 4 Model B and 8 Gb RAM and mounted in outdoor camera boxes. These are then installed on the roof of buildings surrounding the parking lot of a commercial centre. Each camera with the given height, distance and angle from the parking lot, can monitor more than 50 parking spaces. However, due to the presence of occlusions and weather conditions, some parking spots had to be monitored by more than one camera.
- We have assigned a weight value to each pair <parking space-camera>, representing how good is the view of that parking space seen by that camera, following the work *. A high value of this parameter means that the parking space is in the centre of the image and that there is no obstacle between the camera and the parking space. The confidence returned by the CNN for a given parking space is weighted with this value and, in case of a parking space monitored by more cameras, the highest weighted confidence value is selected at server side. In this way, we are able to correct classification errors on parking spaces occluded for some cameras.

*Amato, G., et al. (2017). Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, 72, 327-334.

HYPOTHESIS RESULTS :

- As demonstrated by the inference run, the model generalize well in case of occlusions and new parking lots reaching a mAP of 0.993.

- Our detection inference done by smart cameras avoids high-computing power typical of centralized server, provided that CNN are trained off-line.
- Smart cameras are cheaper than ground sensors as they can monitor more than one parking spot at the same time and have lower installation costs. Complementary activities, such as face/people recognition, or tracking and logging of people and moving vehicles can be potentially added to the device.

- With the detection inference delegated to smart devices, we avoid the bottle-neck of uploading images to the server.

- For more complex decision making and off-road implementations we recommend a different architecture based on cloud, fog computing and VANETs*

*Tang, Chaogang, et al. "Towards smart parking based on fog computing." *IEEE Access* 6 (2018): 70172-70185.