DD2465 Real Time Object Detection, a Case Study

Marco Schouten [1]

Abstract

Real-time object detection is a complex problem and has applications in self-driving vehicles. High accuracy and quick inference are essential for real-time systems. To meet these requirements, I analysed YOLO (*You Only Look Once*), a modern deep neural network used for object detection.

First, I developed the entire pipeline, including data pre-processing, model and training. Second, I analysed the network results on a MakeML dataset suitable for autonomous driving scenarios: (1) Dataset of traffic signs and (2) Dataset of cars and traffic signs.

The results suggest that the training and the network are working correctly, thus demonstrating the implementation of the entire pipeline in the presence of limited computational resources.

[*]Equal contribution [1]Department of EECS, KTH Royal Institute of Technology, Stockholm, Sweden. Correspondence to: Marco Schouten <schouten@kth.se>.

1. Introduction

In this project, I addressed the problem of real-time object detection and investigated solutions based on modern computer vision techniques, such as deep neural networks. Furthermore, an empirical case of YOLOv5 [1] was evaluated on two MakeML datasets.

1.1. Object Detection

Real-time object detection is a task that involves detecting objects in real-time with fast inference while maintaining a basic level of accuracy. Some detected objects are people, cars, chairs, stones, buildings and animals.

This phenomenon seeks to answer two fundamental questions:

1. What is the object? This question seeks to identify the object in a specific image. That is, to determine whether an object (from a predetermined set of possible objects) is present in the image.

2. Where is it located? This question attempts to determine the object's exact position within the image. It is obtained by calculating the boundaries and bounding boxes surrounding the objects.

1.2. Motivation

One motivation for real-time object detection is to assist self-driving cars, i.e. a vehicle that can navigate independently without depending on humans for input (Shreyas et al., 2020).

The automotive industry has made significant progress, introducing new technologies every day. There are various types of autonomous cars, and they are categorised according to their level of automation, ranging from level 1 (assisting with cruise control, such as steering or acceleration) to level 5 (providing complete autonomy) (I. Häring et al.). More specifically, the jump from level 2 to level 3 is substantial from a technological point of view. The difference is that Level 3 vehicles have environmental sensing capabilities and can make conscious decisions, such as accelerating to overtake a slow vehicle. However, they still require human supervision because the driver must remain alert and ready to take control of the system is unable to perform the task.

Real-time object detection plays a crucial role in assisting environmental detection capabilities, e.g. by reading traffic signs or detecting pedestrians or other blunt objects on the road. The system will be able to make accurately informed decisions promptly.

[1] YOLOv5 rocket is a family of object detection architectures and models pretrained on the COCO dataset

1.3. Ethical Implications

Normative Ethics provides three frameworks for addressing ethical questions: consequentialism, Deontology, and Virtue Ethics.

For example, deontology morally critiques actions per se as harm or good according to rights and duties. Real-time object detection can harm citizenship privacy or security if misused and must be avoided. On the other hand, real-time object detection of vital road signs or other objects might give the vehicle enough information to take action just in time to prevent any damage or causality. To this end, road safety will constitute a considerable benefit to society.

1.4. Contribution

A YOLO network was analysed, and an empirical case study was derived from two datasets, (1) Traffic Signals and (2) Cars and Traffic Signals. I integrated the entire ML pipeline, which includes pre-processing the data of an available dataset to make it readable for YOLO. Finally, I analysed the model selection and the search for hyperparameters.

1.5. Outline

In Section 2 I illustrate the main architectures and explore the state of the art for dealing with the classification of objects in real-time, provide an overview of the pros and cons of each strategy and critically choose the most suitable method. In Section 3 I provide technical detail on the YOLO architecture. In Section 4 I describe the experimental approach and analyse the results of our findings. In Section 5 I provide a concluding assessment and suggest directions for future work.

2. Related Work

In this section, I outline the main object detection methods and comment on the strengths of the various methods.

Methods for object detection are divided into neural or non-neural network-based approaches. For non-neural approaches, it is first necessary to define features and then use a technique such as the support vector machine (SVM) to perform the classification. On the other hand, neural techniques can perform end-to-end object detection without precisely defining features and are typically based on convolutional neural networks (CNNs).

2.1. Traditional Computer Vision approaches

A non-neural solution for object detection is the SIFT (scale-invariant feature transform) algorithm, which is used to detect, describe and match local features in images (Lowe,

1999) (Lowe, 2004).

The general idea of SIFT is that for any object in an image, interesting points on the object can be extracted to provide a *feature description* of the object. Therefore by extracting a feature description from a reference image and a test image, we can deduce if the object is present by analysing the similarity of feature descriptions.

In other words, the SIFT key points of the relevant objects are first extracted from a set of reference images and stored in a database. Then features are extracted from a new image, and each feature is compared with this database. The comparison is evaluated by calculating the Euclidean distance between two feature vectors. From the complete set of matches, subsets of key points that agree with the object and its position, scale and orientation in the new image are identified to filter out good matches. These pass rigorous tests: feature detection, matching, Hough transform, Bayesian check for outliers [2].

SIFT features are highly distinctive invariant to image scale and rotation, robust to changes in illumination, noise, and minor changes in viewpoint, making it a good candidate for self-driving applications. Though the Recognition in close-to-real time only for very undersized databases and on modern computer hardware.

In summary, SIFT's accuracy holds its own against more modern algorithms, but it is not the best choice for real-time applications, as it is computationally expensive.

[2]

1. *feature detection* Key positions are defined as maxima and minima of a Gaussian difference function applied in scale space to a series of smoothed and resampled images. Low contrast candidate points and edge response points along an edge are discarded.

2. *Feature matching*: The best candidate for each keypoint is found by identifying its nearest neighbour in the training image keypoint database. The nearest neighbours are the key points with a minimum Euclidean distance from the given descriptor vector. The ratio of the distance of the nearest neighbour to the second-nearest determines whether the match is correct.

3. *Hough transform* identifies feature clusters consistent with the object pose. When feature clusters vote for the same pose, the probability that the pose interpretation is correct is higher.

4. *Outliers are identified by Bayesian testing*, checking the expected number of false matches to the model pose, given the projected size of the model, the number of features within the region and the accuracy of the fit.

2.2. Deep Learning approaches

One could build a detection model on top of a classification model to tackle the problem of object detection using neural approaches. A sliding window algorithm could find the object's position within the image if the object is present. However, the aspect ratio and size of the window are unknown, and it is computationally expensive (especially for neural networks) to find the best values.

In practice, there are two types of mainstream object detection algorithms:

- *Region Proposal* algorithms are implemented in two stages. First, a region of interest where the object is expected to be found in the image is selected. Second, these regions are classified using convolutional neural networks. This solution is slow because we have to run predictions for every selected region. Common networks are R-CNN and Fast(er) R-CNN.

- *Single-shot* algorithms like YOLO (You Only Look Once) (Redmon et al., 2016) and SSD (Single-Shot Detector) [3] (Liu et al., 2016) use a fully convolutional approach in which the network can find all objects within an image in one pass (hence 'single-shot' or 'look once') through the convolutional net.

The region proposal algorithms usually have slightly better accuracy but are slower to run. In contrast, single-shot algorithms are more efficient and have good accuracy; that is what we will focus on in this section.

On the other hand, single shot, fully convolutional approaches are designed for real-time object detection. Faster R-CNN uses a proposal network of regions to create boundary boxes and uses them to classify objects. Although it is considered state-of-the-art in terms of accuracy, the entire process is performed at 7 frames per second, as reported in Figure 2. Far below what is required for real-time processing. The Single-Shot algorithms speeds up the process by eliminating the need for the region proposal network. (Zhang et al., 2018) While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then perform recognition on those regions separately, Single-Shot nets performs all of its predictions with the help of a single fully connected layer.

Methods that use Region Proposal Networks thus end up performing multiple iterations for the same image, while YOLO and SSD gets away with a single iteration.

[3] A great neural solution is Single Shot MultiBox Detector (SSD), an object detection algorithm scoring over 74% mAP (mean Average Precision) at 59 frames per second on standard datasets such as PascalVOC and COCO.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Figure 1: Results on PASCAL VOC 2007 test set from (Redmon & Farhadi, 2016). All timing information is on a Geforce GTX Titan X (original, not Pascal model). the difference in speed (FPS) between region-based (R-CNN) versus single shot (SSD, YOLO) approaches is immediately noticeable. Benchmarks for YOLOv2 and SSD indicate that the real-time object detection strategy for single-shot algorithms is very promising as accuracy results stay high while ensuring high Frames per Second.

3. Method

In this section, I am going to explain the reasons of choice for YOLO to address real-time object detection, its technical details, and the metrics used for the evaluation of the experiments

3.1. Reasons to choose YOLO

The main reason to choose YOLO is that it is faster, running at as high as 45 FPS compared to other algorithms while maintaining good accuracy. It makes it an excellent choice for real-time detection applications such as self-driving vehicles.

In addition, YOLO can capture the context and spatial constraints and it mitigates multiple detections of the same object. (Huang et al., 2018)

Lastly, YOLO learns generalised representations of objects, making it suitable for new environments which are common in driving scenarios.

3.2. Limitations

The main limitation is that it has difficulty detecting small object which appears in a group. It is because YOLO imposes strong spatial constraints on bounding box predictions, as each grid cell only predicts two boxes and can only have one class. This constraint limits the number of neighbouring objects that our model can predict.

In addition, it struggles to generalise to objects with unusual proportions or configurations because it learns to predict bounding boxes exclusively from data. (Liu et al., 2018)

3.3. How does YOLO work?

YOLO is a unified model which frames object detection as a regression problem by simultaneously predicting bounding boxes and class probabilities. A single neural network generates those predictions directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimised end-to-end directly on detection performance. (Du, 2018)

YOLO divides the input image into an $S \times S$ grid of equal size, and if the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object.

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the box is that it predicts. Each bounding box has five predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the box's centre relative to the grid cell's bounds. Finally, the confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities $Pr(Class|Object)$ that quantifies the likelihood of containing an object

At test time YOLO multiply the conditional class probabilities and the individual box confidence predictions:

$$Pr(C_I|O) * Pr(O) * IOU^{truth,pred} = Pr(C_i) * IOU^{truth,pred}$$

Where C_i is the class, O is an object, and IOU is the Intersection over Union.

This process dramatically lowers the computation as both detection and recognition are handled by cells from the image.

However, it brings forth a lot of duplicate predictions due to multiple cells predicting the same object with different bounding box predictions. YOLO uses Non-Maximal Suppression to deal with this issue: it suppresses all bounding boxes with lower probability scores. YOLO achieves this by first looking at the probability scores associated with each decision and taking the largest one. Following this, it suppresses the bounding boxes having the most prominent Intersection over Union with the current high probability bounding box. This step is repeated till the final bounding boxes are obtained.

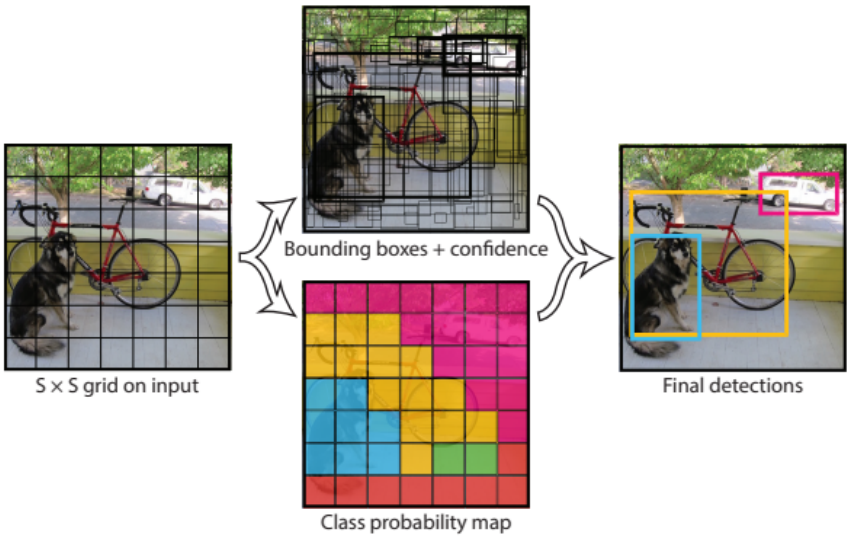

Figure 2: YOLO models detection as a regression problem. It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor

3.4. Metrics

The evaluation metrics of the object detection problem follow precise definitions agreed by the research community. Hereunder will be listed the primary performance metrics.

3.5. Intersection of Union

Intersection Over Union (IOU) evaluates the overlap between the ground truth bounding box and a predicted bounding box. If the percentage of overlapping area is greater than a threshold, then the detection is valid. Consequently, IOU offers a quantitative metric to determine if detection is valid (True Positive) or not (False Positive). Common threshold values are set at $50\%, 75\%95\%$

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

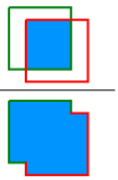

Figure 3: Visual representation of Intersection over Union between a ground truth bounding box (in green) and a detected bounding box (in red).

3.6. Precision x Recall curve

An object detector of a particular class is considered good if its precision stays high as recall increases. If you vary the confidence threshold, the precision and recall will still be high. Another way to identify a good object detector is to look for a detector that can identify only relevant objects (0 False Positives = high precision), finding all ground-truth objects (0 False Negatives = high recall).

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

3.7. Average Precision

Another way to compare the performance of object detectors is to calculate the area under the curve (AUC) of the Precision x Recall curve. In practice, AP is the precision averaged across all recall values between 0 and 1.

4. Experiments

In this section, I explain the setup and training results for a YOLO model. The experimental setup was tested using a YOLOv5 pre-trained on the COCO dataset.

Model	size(pixels)	mAP 0.5 : 0.95	V100	FP16
YOLOv5small	640	37.2	6.4ms	14MB

Table 1: Pre-trained Model specifications(on COCO)

4.1. Setup

To carry out the The experiments involving the YOLO model (YOLOv5s [4]) described in Section 3 were carried out on the following public domain datasets (Müller & Dietmayer, 2018):

- MakeML Road Signs, a collection of road signs of four different types: Traffic Light, Stop, Speed limit, Crosswalk. [5]
- Cars and Traffic Signs, a larger collection of items including Car, Truck, Person, Traffic signs, Traffic Light etc. This second dataset was more complex due to the larger variety in the data. [6]

In both cases, the dataset contained two file types, the image and their respective annotations, i.e. the information related to the bounding boxes and the class. Both datasets were formatted according to the popular PASCAL VOC XML format.

However YOLO model needs the dataset in a specificed txt file where for each image:

- One row per object
- Each row is a tuple: { class, x_{center}, y_{center}, width, height }.
- Box coordinates must be normalized by the dimensions of the image (i.e. have values between 0 and 1)
- Class numbers are zero-indexed (start from 0).

To accommodate this requirement, I developed a script that converted XML PASCAL annotations to the TXT format required by YOLO.

The dataset was partitioned into Training, Validation, and Test sets with a split of 80%, 10%, and 10% of the data, respectively.

The best hyperparameters settings is found at: Initial learning rate of 0.01 (SGD=1E-2, Adam=1E-3); final OneCycleLR learning rate (lr0 * lrf), where lfr is 0.2; momentum: 0.937,; IoU training threshold 0.20; 3 warmup epochs.

[4]Pretrained YOLOv5s available at https://github.com/ultralytics/yolov5

[5]Publicly Available at https://makeml.app/datasets/road-signs

[6]Publicly Available at https://makeml.app/datasets/cars-and-traffic-signs

The network comprises two blocks: the backbone and the head. Model Backbone (CSP-Darknet5) is mainly used to extract essential features from the given input image. The YOLO head described in section 3 is mainly used to execute the detection. I Trained the model on NVIDIA GT1050, and due to memory limitations, the maximum minibatch size possible was 16.

4.2. Results

The model's performance over a dataset is evaluated on a validation set computing the loss. The quantitative results represent averages over three training sessions. However, it is enough to show how the network behaves and draw conclusions.

YOLOV5S can achieve good results. Being trained for 20 epochs with each dataset warmup of 3 epochs, the model can detect the object with high recall and precision the data while capturing the most important features (see Result Table). We have high enough recall and precision in both datasets, which testifies that the implementation is working as expected on the tested datasets. That is, the regression output of the fine-tuning correctly inferring the bounding boxes and location and object classifications.

The results were good in both datasets considering the training was limited to a smaller number of epochs due to hardware limitations.

Regarding system settings, the training time was under 2 hour for the Road Sign dataset, and under 3 hours for the Larger dataset.

Lastly, a few examples of inference on the test sets are shown at the end of the report, precision and recall results are great, in particular for stop signs.

5. Conclusion

Different metrics are inspected: Precision, Recall, Averages, Loss, and time. Yolo proved to be a good real-time object detection.

- By analyzing inference pictures, the Road Signal dataset had the best results (comparing for Precision and Recall). It can probably be explained by the fundamental difference that the other dataset had many more classes and was a superset of the first one.
- By analyzing the Loss functions, we can see that the

Dataset	Precision	Recall	mAP 0.5	mAP 0.5:0.95
Road Signal	0.9857	0.949	0.967	0.726
Cars and Traffic Signs	0.911	0.909	0.942	0.699

Table 2: Metric Result. The table reports good metrics for both datasets. The Cars and traffic sign dataset had more classes and the perfromances were less good compared to a dataset with less classes.

Figure 4: *Object detection Inference part 1* in the presence of big enough objects such as road traffic signs, the network could correctly identify multiple instances. It was behaving better than expected, as identifying multiple instances is part of the expected shortcomings of YOLO. Considering the limited hardware at my disposal, the results were better than expected. On the other hand, the network still makes some mistakes. As you can see in the left-most image, the stop signal is identified as a crosswalk.

training was successful, and the overfitting was sufficiently regularised to make the network correctly detect objects in the test set.

- By analyzing time performance, the training was faster in the smaller dataset, but the amount was not too large. Since the difference in size was not that big.
- Warm-up throughout experiments showed to be significant for avoiding too much regularisation of the result, which causes the model to lack detection abilities.
- Regarding the learning rate, we noticed that a successful training was to be set to a lower value than with standard (by a factor of 2); perhaps this comes from the smaller dataset.

5.1. Future Developments

This research showed the great potential of a single shot network as a real-time detection model. It would be fascinating to compare alternative networks such as SSD and discriminate which induces the best results for specific problem instances. Even more compelling would be to test how this system performs for much larger datasets, to test the scaling of the system. Lastly, even within YOLO, there are many variations based on the head's size. It would be great to inspect and compare them to find the best ad-hoc network configuration for a particular dataset. And then maybe combining inferences of various nets working in parallel to have the highest confidence possible in the predictions.

Figure 5: *Object detection Inference part 2*. Here it is easy to seet that even small objects such as trafficlights , were correctyly detected. Hoeever performances on crassworks and speedlimits were not as good, due to the fact that the database was skewed, meanring it had less istances of the objects as reported in the label distribtuon table

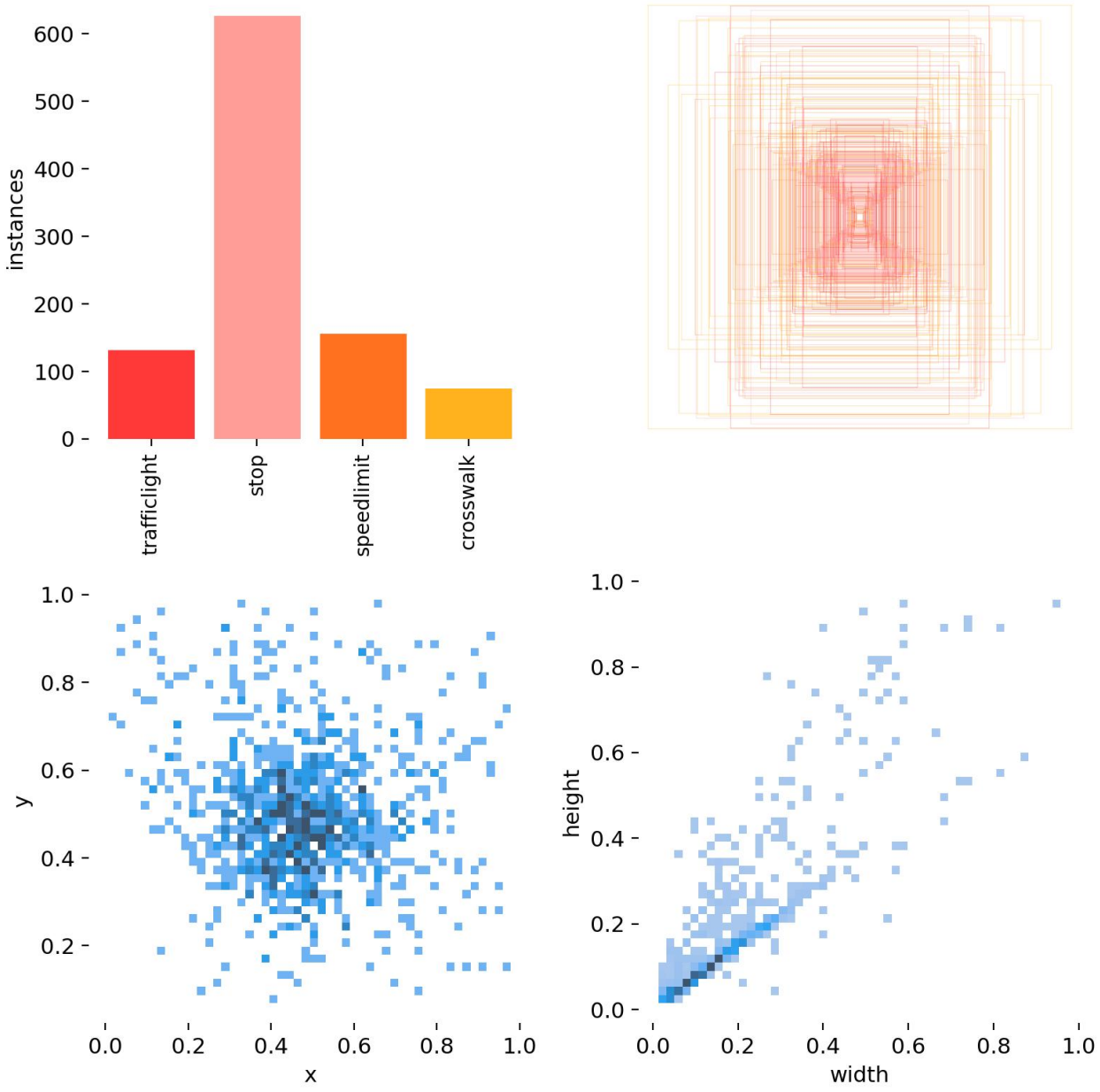

Figure 6: *Label distribution in Road Sign Dataset.* As you can see from the histogram, there were many more stop signal instances compare to the other classes, and the correct detections of crosswalks were less frequent.

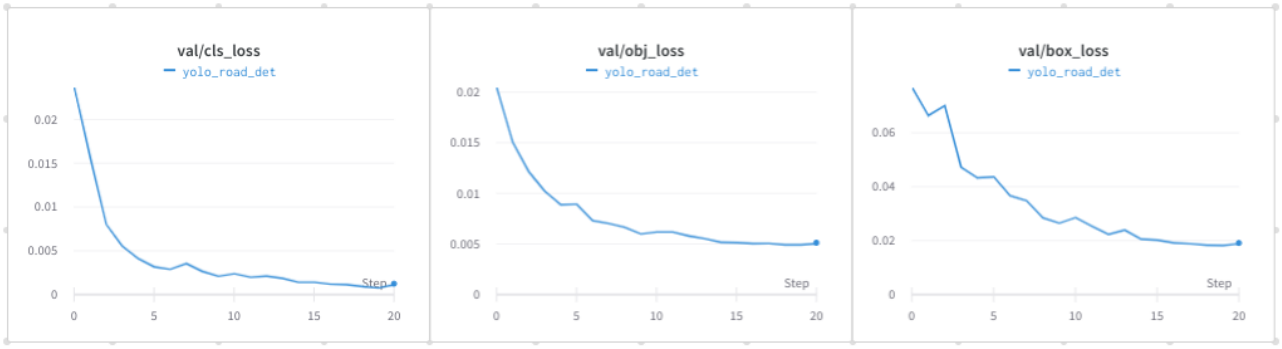

Figure 7: *Validation Loss of Yolov5s.* (1) CLs is a loss that measures the correctness of the classification of each predicted bounding box: each box may contain an object class, or a ”background”. This loss is usually called cross entropy loss. (2) obj is equal to one when there is an object in the cell, and 0 otherwise. (3) Box loss measures how ”tight” the predicted bounding boxes are to the ground truth object

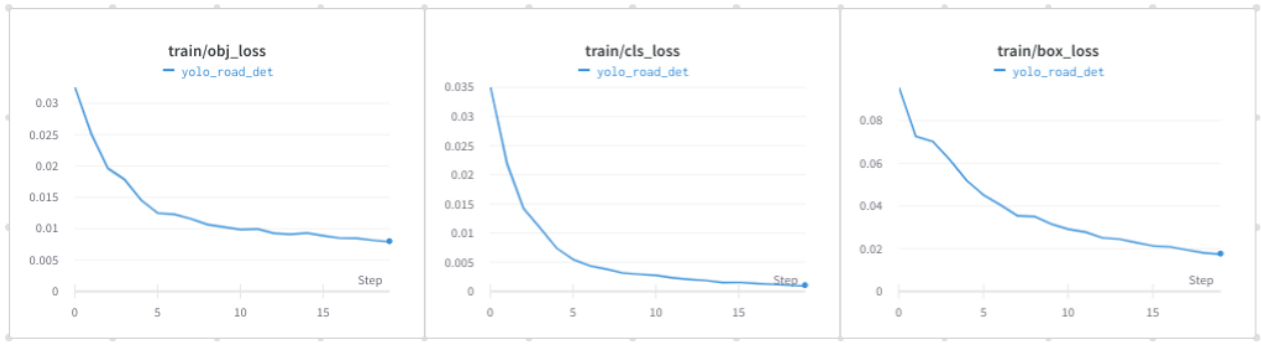

Figure 8: *Training Loss of Yolov5s.* (1) CLs is a loss that measures the correctness of the classification of each predicted bounding box: each box may contain an object class, or a ”background”. This loss is usually called cross entropy loss. (2) obj is equal to one when there is an object in the cell, and 0 otherwise. (3) Box loss measures how ”tight” the predicted bounding boxes are to the ground truth object

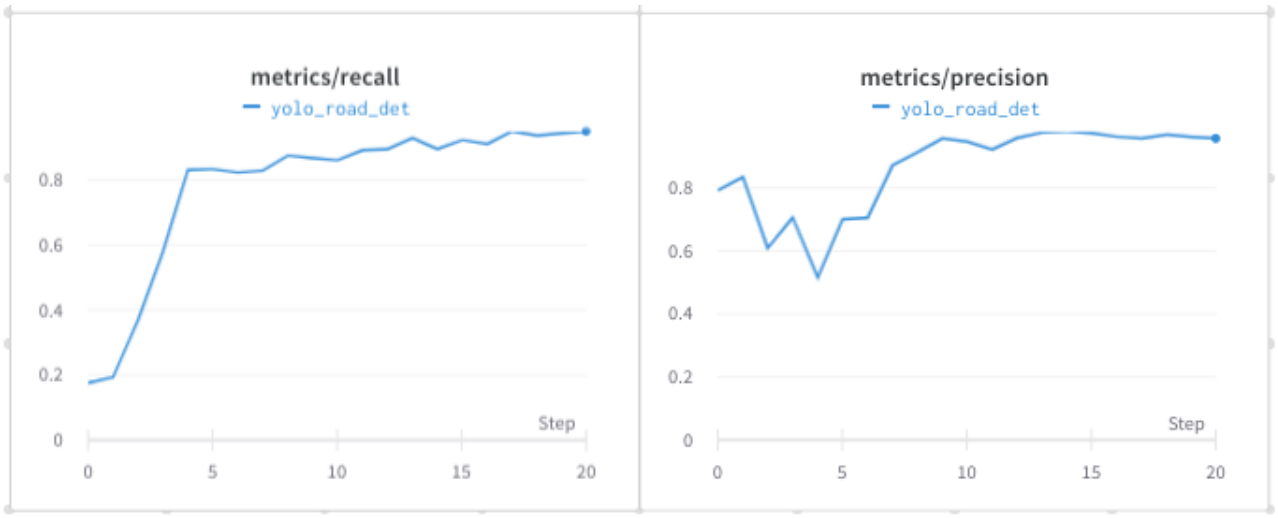

Figure 9: Recall and Precision are high indicating that the training was sufficient, and the object detection is working as expected. The regularisation terms are working and there is not significantly over-fitting.

References

Du, J. Understanding of object detection based on cnn family and yolo. In *Journal of Physics: Conference Series*, volume 1004, pp. 012029. IOP Publishing, 2018.

Huang, R., Pedoeem, J., and Chen, C. Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 2503–2510. IEEE, 2018.

I. Häring, Florian Lüttner, Andreas Frorath, Miriam Fehling-Kaschek, Katharina Ross, T. Schamm, Steffen Knoop, Daniel Schmidt, Andreas Schmidt, Yang Ji, Zhengxiong Yang, A. Rupalla, Frank Hantschel, Michael Frey, Norbert Wiechowski, C. Schyr, D. Grimm, M. Zofka, and A. Viehl. Framework for safety assessment of autonomous driving functions up to SAE level 5 by self-learning iteratively improving control loops between development, safety and field life cycle phases. *2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP)*. doi: 10.1109/iccp53602.2021.9733699.

Liu, C., Tao, Y., Liang, J., Li, K., and Chen, Y. Object detection based on yolo network. In *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 799–803. IEEE, 2018.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. SSD: Single Shot MultiBox Detector. volume 9905, pp. 21–37. 2016. doi: 10.1007/978-3-319-46448-0_2.

Lowe, D. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pp. 1150–1157 vol.2, September 1999. doi: 10.1109/ICCV.1999.790410.

Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000029664.99615.94.

Müller, J. and Dietmayer, K. Detecting traffic lights by single shot detection. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 266–273. IEEE, 2018.

Redmon, J. and Farhadi, A. YOLO9000: Better, Faster, Stronger, December 2016.

Redmon, J., Divvala, S. K., Girshick, R., and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, June 2016. doi: 10.1109/cvpr.2016.91.

Shreyas, V., Bharadwaj, S. N., Srinidhi, S., Ankith, K. U., A. B. Rajendra, A. B. Rajendra, and Rajendra, A. B. Self-driving Cars: An Overview of Various Autonomous Driving Systems. pp. 361–371, January 2020. doi: 10.1007/978-981-15-0694-9_34.

Zhang, S., Wen, L., Bian, X., Lei, Z., and Li, S. Z. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4203–4212, 2018.