MOD-CL: Multi-label Object Detection with Constrained Loss

Sota Moriyama
National Institute of Informatics
Tokyo, Japan
sotam@nii.ac.jp

Koji Watanabe
National Institute of Informatics
Tokyo, Japan
kojiwatanabe@nii.ac.jp

Katsumi Inoue
National Institute of Informatics
Tokyo, Japan
inoue@nii.ac.jp

Akihiro Takemura
National Institute of Informatics
Tokyo, Japan
atakemura@nii.ac.jp

Abstract

We introduce MOD-CL, a multi-label object detection framework that utilizes constrained loss in the training process to produce outputs that better satisfy the given requirements. In this paper, we use MOD_{YOLO}, a multi-label object detection model built upon the state-of-the-art object detection model YOLOv8, which has been published in recent years. In Task 1, we introduce the Corrector Model and Blender Model, two new models that follow after the object detection process, aiming to generate a more constrained output. For Task 2, constrained losses have been incorporated into the MOD_{YOLO} architecture using Product T-Norm. The results show that these implementations are instrumental to improving the scores for both Task 1 and Task 2.

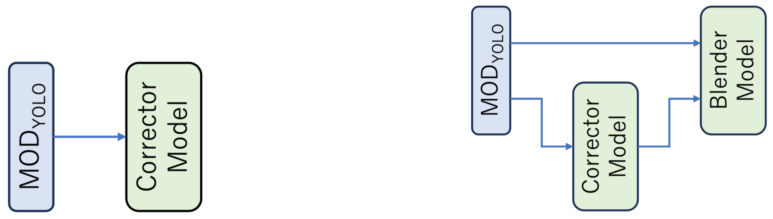
1 Introduction

Object detection is a critical computer vision task that aims to identify the precise locations of objects in images or videos [1]. Due to the nature of this task, object detection has received a lot of attention in the context of autonomous driving. However, with autonomous driving, there is a need to have further information about the action the object is taking (action detection). On top of this, there has been some extensive research about making the outputs satisfy some given requirements [2]. These are all crucial to the development of autonomous driving.

In this paper, we introduce MOD-CL, a multi-label object detection framework that utilizes constrained loss in the training process to better satisfy the requirements of the actions. Specifically, we develop a model called MOD_{YOLO} that is based on YOLOv8 [3], a state-of-the-art object detection model released by Ultralytics. Additionally, we introduce two distinct models and training procedures to accommodate two different scenarios: (1) limited labeled data, and (2) the output of labels that satisfy the requirements. These will be referred to as Task 1 and Task 2 from here.

2 YOLOv8 for Multi-labeled Object Detection

As the original YOLOv8 only supported single labels per bounding box, we have modified the program to support multiple labels. To accomplish this, we used n-hot vectors instead of the original



(a) Stage 1 of training (b) Stage 2 of training

Figure 1: MOD_{YOLO} with Corrector-Blender Model. Green blocks represent the parts being trained.

one-hot vectors as ground truths to train the model to have bounding boxes that have high confidence scores for multiple labels. Numerous parts of the original YOLOv8 model, including the input sequence and output sequences, have been modified. However, we will only touch on parts that directly affect the performance of the overall model, namely the Non-Maximum Suppression (NMS) algorithm that is used before outputting the predictions.

In the original YOLOv8 model, the outputs are produced through the use of NMS with regards to the confidence scores and IOU of each label for each bounding box. On the other hand, in our approach, we focused on outputting the bounding boxes with respect to the confidence scores of the agent labels. Specifically, the modifications done are as follows:

- **Bounding Box Thresholding**: Only bounding boxes that have confidence scores for agent labels above the threshold are used.
- **Agent-wise NMS**: Excessive bounding boxes were reduced with the use of the NMS algorithm with respect to the bounding boxes and confidence scores of the agent labels.

This not only allows us to output multiple labels for every bounding box, but it also allows us to satisfy one of the pre-defined requirements: to have at least one agent label included.

As each of the bounding boxes has one label that corresponds to the type of agent, we focus on filtering with respect to the agent labels. This prevents the model from violating the requirement for the agents—to have at least one agent label included in the output. On top of all of the modifications, we also used a SORT algorithm, BoT-SORT [4] (already implemented in YOLOv8), to enhance our output accuracy.

We will refer to this variation of the YOLOv8 model with the above modifications, MOD_{YOLO} (Multi-labeled Object Detection based on YOLO).

3 Task 1

3.1 Method

Task 1 is focused on training using a partially labeled dataset. For this purpose, we focused on employing a semi-supervised learning method to fully utilize the labeled and unlabeled parts of the dataset. We break our training process down into two parts: (1) supervised learning of MOD, and (2) semi-supervised training on the Corrector and Blender models.

As the first half of the training is done in a traditional fashion, we will only touch on the latter half in this section. An abstract figure of our method is shown in Figure 1. In our method, we introduced two new models: the Corrector Model and Blender Model. The semi-supervised training process is split into two parts: (1) unsupervised learning on the corrector model, and (2) supervised learning on the corrector and blender models. The two stages of training are done in the same order every training epoch, to reduce the risk of overfitting to either the labeled or unlabeled portions of the dataset.

In Stage 1 of the training process (shown in Figure 1a), the corrector model is being trained with the unlabeled parts of the dataset in an unsupervised manner. The loss being used in this stage is based on the constrained loss shown in the original ROAD-R paper [2]. The loss is set as follows:

$$\mathrm{L}_{S1} = \frac{1}{243}\sum_{i=1}^{243} t(r_i) \tag{1}$$

Table 1: Comparison of models for Task 1

Model	Frame-mAP@0.5
Baseline Model	0.18
MOD$_{\text{YOLO}}$	0.2633
MOD$_{\text{YOLO}}$ with Corrector-Blender Model	**0.2662**

Table 2: Comparison of models for Task 2

Model	Precision@0.5	Recall@0.5	F1-Score@0.5
Baseline Model	0.49	0.34	0.40
MOD$_{\text{YOLO}}$	0.6769	0.4430	0.5355
MOD$_{\text{YOLO}}$ with Constrained Loss	**0.7057**	**0.5405**	**0.6122**

Here, r_i corresponds to the ith requirement, and $t(r_i)$ corresponds to the fuzzy logic relaxations of r_i. Furthermore, we focused on using Product T-Norm for the fuzzy logic relaxation, and the predicted scores used in this loss are solely from the corrector model.

In Stage 2 of the training process (shown in Figure 1b), the corrector and blender models are being trained with the labeled parts of the dataset in a supervised manner. The loss is set as follows:

$$\mathrm{L}_{S2} = 10 \times \mathrm{L}_{S1} + \text{BCELoss}(\hat{\boldsymbol{y}}_C, \boldsymbol{y}) + \text{BCELoss}(\hat{\boldsymbol{y}}_B, \boldsymbol{y})$$

Here, $\hat{\boldsymbol{y}}_C$ and $\hat{\boldsymbol{y}}_B$ correspond to the output predictions given by the corrector model and the blender model.

3.2 Results

The results for the comparison of models are shown in Table 1. From the results, we can see that using the Corrector Model along with the Blender Model can have a positive impact on the overall performance. This shows that having minor changes to label confidence scores with respect to the requirements is important, even under normal object detection circumstances. Moreover, this demonstrates the positive impacts of using the unlabeled parts of the dataset to learn the requirements in an unsupervised manner.

4 Task 2

4.1 Method

Task 2 is focused on using the full dataset to train and finally outputting a set of labels that fully satisfy the requirements. To train the model that outputs confidence scores that directly lead to the correct output with the use of solvers, we use the constrained loss shown in Equation (1). However, YOLO outputs confidence scores for each of its anchor points (total of $W \times H$ anchor points), making calculating the loss very expensive, both computation time-wise and resource-wise. Therefore, we calculate the constrained loss on anchor points that have at least one label with confidence scores above 0.5. Not only does this allow for less computation time and resources, but it also reduces the risk of trying to output high confidence scores for every bounding box, as the requirements do penalize empty label sets.

After the training of the model, we used MaxHS [5], a Partial Weighted MaxSAT solver, in an identical way to the original ROAD-R paper [2]. This allowed us to output labels that are guaranteed to satisfy the given requirements.

4.2 Results

The results for the comparison of models are shown in Table 2. From the results, we can see that using constrained loss has a big effect on the overall performance of the model in terms of every metric. This shows that in models, learning to satisfy the requirements is needed if we were to use complete solvers to output sets of labels that are guaranteed to satisfy the requirements.

Another reason for the high scores is thought to be because of the agent-wise thresholding/NMS explained in Section 2. As at least one of the confidence scores of agent labels given to the MaxSAT solver is guaranteed to be above the threshold, it allows the solver to essentially not care about one of the requirements. Although the amount of leverage this method gives is not verified, we believe that there is a certain effect as the scores of MOD_{YOLO} itself will place in 5th place of Task 2.

5 Conclusion

In this paper, we introduced MOD-CL, a framework that focuses on using constrained losses with multi-labeled object detection. We used MOD_{YOLO}, a model based on the SotA object detection model YOLOv8, and implemented constrained losses in different forms to suit the separate tasks. In Task 1, we focused on modifying the outputs of YOLOv8 to give an output that satisfies the requirements more. From the results, we found that modifying the results in a particular way has a positive impact on the overall performance of the model. In Task 2, we focused on learning with constrained loss included to give outputs that lead to correct answers with the use of Weighted Partial MaxSAT solvers. The results show that the use of the constrained loss has a positive effect on the overall performance, suggesting its effectiveness in practical applications.

Acknowledgments and Disclosure of Funding

This work has been supported by JSPS KAKENHI Grant Number JP21H04905 and JST CREST Grant Number JPMJCR22D3.

References

[1] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object Detection in 20 Years: A Survey. *Proc. IEEE*, 111(3):257–276, 2023.

[2] Eleonora Giunchiglia, Mihaela Catalina Stoian, Salman Khan, Fabio Cuzzolin, and Thomas Lukasiewicz. ROAD-R: the autonomous driving dataset with logical requirements. *Mach. Learn.*, 112(9):3261–3291, 2023.

[3] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, January 2023.

[4] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[5] Randy Hickey and Fahiem Bacchus. Speeding Up Assumption-Based SAT. In Mikolás Janota and Inês Lynce, editors, *SAT 2019*, volume 11628 of *Lecture Notes in Computer Science*, pages 164–182. Springer, 2019.