CS 5804 Intro to AI (Spring 2023) Final Project Team 8:

Deep Learning-Based High-Accuracy Traffic Sign Detection

Kaiyi Chen
Virginia Tech, kennychen@vt.edu
Xiao Guo
Virginia Tech, kevinguo2003@vt.edu
Chujia Chen
Virginia Tech, cjchen@vt.edu
Shihong yang
Virginia Tech, shihong@vt.edu

Abstract

Our mini-project proposes a deep learning-based approach for traffic sign detection, which is a crucial task for autonomous driving systems. German Traffic Sign Detection Benchmark dataset is used and a YOLOv7 model is trained to detect traffic signs of different shapes, colors, and sizes, even under varying lighting conditions. The YOLOv7 made some significant contributions to the YOLO network and training routines such as extended efficient layer aggregation, model scaling techniques, planned re-parameterization, and lead head guided label assigner. Our project evaluates the YOLOv7 model using precision, recall, and Mean Average Precision (mAP) metrics and demonstrates its high accuracy and real-time performance in detecting traffic signs.

Keywords and Phrases: Deep Learning, YOLO, Traffic Sign Detection, Computer Vision

1 Problem Description

Traffic sign detection is an important task for autonomous driving systems because it provides the necessary information for navigation and safety. It is an important task for autonomous driving systems. The particular difficulty of detecting the changes in sign size, shape, color, occlusion, and illumination makes it a hot topic in the computer version domain. Moreover, traffic sign detection also requires high accuracy and real-time performance, which is usually a natural shortage of object detection models [1].

2 Dataset

The dataset is German Traffic Sign Detection Benchmark (GTDRB) [2], which including 900 photos with traffic signs. The images in this dataset contain anywhere from zero to six traffic signs. These signs are grouped into: prohibitory, danger, mandatory and others. This enables us to train our algorithm to recognize a wide variety of traffic signs that a driver might encounter. In addition, the photos' traffic sign sizes range from 16x16 to 128x128 pixels. This means that our algorithm will be able to detect smaller signs in larger images because it has been trained to recognize traffic signs of

different sizes. Furthermore, the dataset includes images that traffic signs from every perspective and under every lighting condition. This will allow our algorithm to learn how to identify traffic signs regardless of their orientation or the lighting conditions in which they appear.

To facilitate training, the original dataset has been converted to the YOLO format [3]. This format is widely used for object detection tasks and will allow us to efficiently train our algorithm on this dataset. Specifically, the images are in *.jpg format and have corresponding *.txt files with YOLO annotations. These annotations include bounding box information in the format of [Class Number] [center in x] [center in y] [Width] [Height]. This will help our algorithm learn where the traffic signs are located in the images and will improve its ability to detect and classify traffic signs accurately.

3 Proposed Approach

3.1 Methods

3.1.1 Load the GTRSB dataset

Output the dataset through the terminal command in the YOLOv7 - PyTorch format [3]. Use the code snippets provided in the tutorial to load the data and the YOLOv7 model baseline for fine-tuning.

3.1.2 Set up the environment for training

Install the required packages and run the necessary commands to set up the environment for training. This will include installing PyTorch, CUDA, and other required libraries.

3.1.3 Train the model

Run the YOLOv7 model on the training data and validate it on the validation and test sets. Fine-tune the model by adjusting the hyperparameters, such as the learning rate and batch size.

3.1.4 Test the model

Create our own qualitative test by testing the model on a new highlight reel, and evaluate the performance of the model on new data.

3.1.5 Improve the model

Allow the model to learn from a more diverse set of examples and improve its performance. Experiment with different architectures and hyperparameters to improve the model's accuracy.

3.2 Innovative Strategies

3.2.1 Extended efficient layer aggregation

It improves the efficiency of the YOLO network's convolutional layers in the backbone for faster inference. The YOLOv7 built on previous research and proposed E-ELAN, an extended version of the ELAN computational block. This method enhances feature learning, parameter utilization, and calculation efficiency by increasing the cardinality of added features through group convolution and combining features of different groups.

3.2.2 Model scaling techniques

It considers the network's depth, width, and resolution trained on. In YOLOv7, the authors scale the network depth and width while concatenating layers together. To maintain the optimal model architecture while scaling for different sizes, they propose scaling only the depth in a computational block and performing corresponding width scaling for the remaining transmission layers. This technique allows for optimal model architecture while scaling for different sizes.

3.2.3 Planned re-parameterization

It improves the performance of the model. This approach involves replacing the RepConv layer with RepConvN when the previous layer has residual or concatenation connections, but no identity connection. This change helps to prevent the model from getting stuck in local optima, leading to better accuracy and efficiency. Overall, the planned re-parameterization technique is a promising new approach for improving the performance of the YOLOv7 model.

3.2.4 Lead head guided label assigner

This method optimizes the labels for both the lead head and auxiliary head simultaneously by using lead head predictions and ground truth data. The lead head guided label assigner is a refinement of the usual independent label assigner and is optimized to achieve better performance. The proposed method also includes a coarse-to-fine implementation, which allows for more accurate labeling. The details of the implementation and constraint design are discussed in the Appendix. Overall, this new label assigning method is expected to improve the performance of the YOLOv7 model.

3.3 Running Instructions

Link to our code:
https://drive.google.com/file/d/1vMzP2HcKIIxEuk3IRGptevjRrgDBI3aa/view?usp=sharing
Please download the zipfile named yolov7-main.zip to run our project.
Requirement：cuda, requirement.txt

The parameters have been configured. After putting the files you need into the folder, change the file directory in detect.py to the address of the folder you need to run. After running detect.py, you can find the picture or video content marked by the model for your data in the running record.

Evaluation

To understand our approach's performance, we record the validation accuracy metrics over training. We adopted the commonly-used Precision and Recall metric. In addition, YOLO provided a special metric called Mean Average Precision (mAP) over a certain Intersection Over Union (IOU) that was specifically designed for the object detection task. We recorded the trend of precision, recall, mAP @ 0.5, and mAP @ [0.5, 0.95] over the 300 training epochs. The results are shown in Figure 1 and Figure 2. The performance trend is increasing with the training time perfectly while suffering from a little initial oscillation. Still, we observe that the majority of training is done by the first half of the training process or 150 epochs. The improvement of performance becomes relatively slow for the last 100 epochs compared to the first one as the model weights are almost settled.

We also present our best performance throughout the training process in Table 1. As the highest precision and mAP are all promising, we believe that our model has done the traffic sign detection task in an accurate manner.

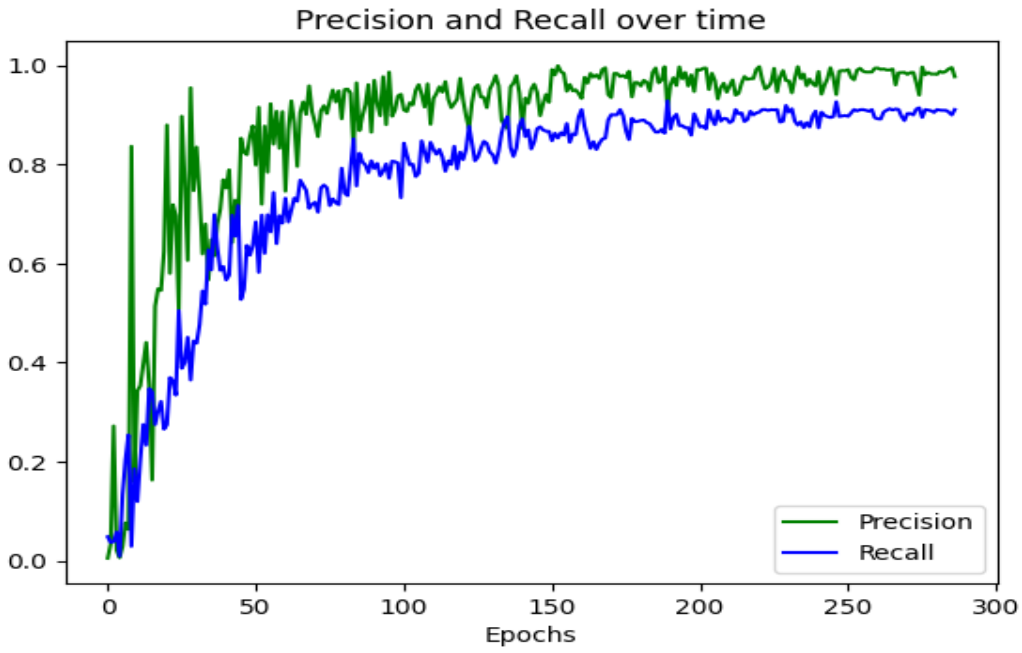

Figure 1: Precision and Recall over the training period.

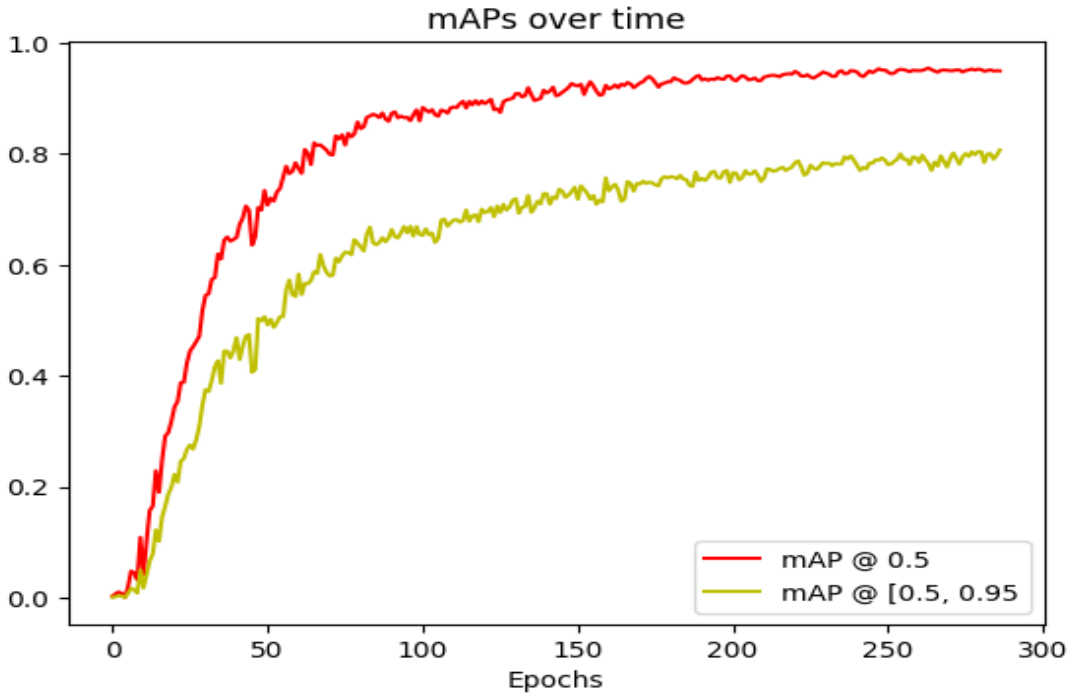

Figure 2: mAP at 0.5 IOU and mAP at IOU range from [0.5, 0.95] over the training process

Precision	Recall	mAP @ 0.5	mAP @ [0.5, 0.95]
0.9995	0.9286	0.9550	0.8072

Table 1: Highest evaluation score over the training process

4 Discussion and Future Work

Throughout the project, we have encountered multiple difficulties. The process of solving this problem also brings new thoughts regarding our project. Firstly, we found that there is a significant difference in the time it takes to run the model on different hardware environments. Our experiments are done on a local GTX1080 graphic card and take more than 6 hours. It also requires a lot of effort to set up the Cuda environment so that the model can utilize the GPU acceleration. However, we believe that switching to a cloud computing platform will take less effort to set up and also provide better computation power. The second lesson we learned is the importance of the dataset's quality. As we are classifying the traffic signs, which requires a very high accuracy due to the nature that it's life-related, our training dataset should cover as much as extreme conditions as possible. While the model itself does support the transfer of learned knowledge to unknown climate conditions, it's always better to cover those conditions in the dataset. Lastly, we learned that our model, while technically feasible, is

still far away from real deploying because we need to increase the inference speed. To be specific, it's essential to identify traffic both fast and accurately because cars don't have much time to react to the changing environment. We may need to adjust the internal architecture of the network to increase its running speed so that it can respond to the input images in a timely manner.

Based on the lessons we learned, we proposed several directions for future work as follows:

- Use other datasets: In future work, we plan to use other datasets, such as CSUST Chinese Traffic Sign Detection Benchmark (CCTSDB) [4]. CCTSDB provides larger and more detailed traffic sample sets. We hope training on this dataset will make our model more robust when facing different challenging conditions.
- Utilize cloud computing platforms: As we found that training on a customer-level GPU takes a significant amount of set-up and time effort, we plan to utilize cloud computing platforms such as Colab and AWS to speed up the training process. By doing so, we could have more time to tune the model for better performance.
- Compare other YOLO models' performance: We will train different versions of the YOLO model with the same dataset and compare their performance. We expect to see how the change in architecture will affect the final task performance.
- Segment Anything Model (SAM) for object detection: Our task can be divided into two parts: object detection and classification. By deploying the newest-published SAM model from meta, we could tremendously reduce the time it takes to segment the target traffic sign from its surround environment, thus leaving us more time to inference its category.

5 Conclusion

To sum up, our experiment has shown that using the YOLOv7 object identification model for highly accurate traffic sign detection is both feasible and effective. On the GTRSB dataset, the model has displayed exceptional performance. This suggests that even under difficult circumstances like low illumination or fog, the model was able to recognize and categorize traffic signs effectively. The study has demonstrated the bright future of applying computer vision models to boost the capabilities of autonomous driving systems and increase road safety. More thorough and quicker traffic sign detecting systems may result from more research and development in this field, which has the potential to change how people drive their cars completely.

Reference

[1] Sichkar V. N. Effect of various dimension convolutional layer filters on traffic sign classification accuracy. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 2019, vol. 19, no. 3, pp. 546–552. DOI: 10.17586/2226-1494-2019-19-3-546-552 (Full-text available on ResearchGate here: Effect of various dimension convolutional layer filters on traffic sign classification accuracy.
[2] Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011, July). The German traffic sign recognition benchmark: a multi-class classification competition. In The 2011 international joint conference on neural networks (pp. 1453-1460). IEEE.
[3] Wang, Chien-Yao. "[2207.02696] YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." arXiv, 6 July 2022, https://arxiv.org/abs/2207.02696. Accessed 4 May 2023.
[4] Zhang, J., Zou, X., Kuang, L. D., Wang, J., Sherratt, R. S., & Yu, X. (2022). CCTSDB 2021: a more comprehensive traffic sign detection benchmark. Human-centric Computing and Information Sciences, 12.