Applying YOLOv5s for Traffic Sign Recognition

Balashova Ekaterina
VinUniversity
Hanoi, Vietnam
21ekaterina.b@vinuni.edu.vn

1. Introduction

Traffic sign recognition is a crucial component of autonomous vehicles and advanced driver-assistance systems (ADAS). Identifying them correctly and timely is crucial to avoid accidents on the road and ensure that the vehicle abides by the traffic regulations.

However, a major challenge in this project was the lack of suitable public datasets. The only dataset I was able to find included bounding box annotations, but it was better suited for classification rather than object detection, as most of the images were close-ups of traffic signs. This posed a significant hurdle, as traffic signs in real-world scenarios appear at various scales and positions—a feature missing in the dataset.

The following approach was adopted to address the limitations and attempt to build a traffic sign detector:

- Extensive data augmentation, including oversampling and synthetic image generation, to compensate for dataset weaknesses.
- YOLOv5s, a widely used real-time object detector, for its speed-performance tradeoff and ability to generalize well.

2. Proposed approach

2.1. Addressing problems with data

2.1.1 Class imbalance – quantile-based oversampling

One of the big issues with the dataset was a complicated case of class imbalance: some classes had upwards of 2000 images, and some had as little as 200-300. Oversampling all classes to the same number of instances would likely lead to overfitting on rare classes, so a quantile-based strategy was used to find balance between providing enough representation for rare classes and avoiding excessive duplication:

- Classes with very few samples (below Q1) were strongly oversampled to reach Q2.
- Moderately underrepresented classes (between Q1 and Q2) were slightly oversampled (to (Q2 + Q3) / 2) to improve representation.
- Classes above Q2 were left unchanged.

2.1.2 Generalization – synthesizing new images

Because the dataset consisted mainly of cropped close-up images of traffic signs, it lacked the variations that real street images would have, like different scales and positions within a scene. To address this, synthetic images were generated by rescaling traffic signs to be smaller and embedding them in larger canvases to simulate real-world distances. The positions of signs on canvases were randomized.

2.2. The model

YOLO (You Only Look Once) is one of the most popular and effective deep-learning-based object detection models. It was chosen for this project because of its real-time performance and efficiency. Unlike two-stage detectors like Faster R-CNN, YOLO operates in a single pass, making it well-suited for real-time applications, such as traffic sign detection in autonomous vehicles [4].

Among the YOLO model family, YOLOv5 was selected because:

- It supports easy model fine-tuning with pre-trained weights.
- Though not current state of the art, it offers great speed-accuracy balance [2].
- The YOLOv5s specifically was chosen for its efficiency and lower computational requirements, making it suitable for training on a mid-range GPU (NVIDIA GTX 1650, 12GB VRAM) [3].

The structure of YOLOv5 can be seen on Figure 1.

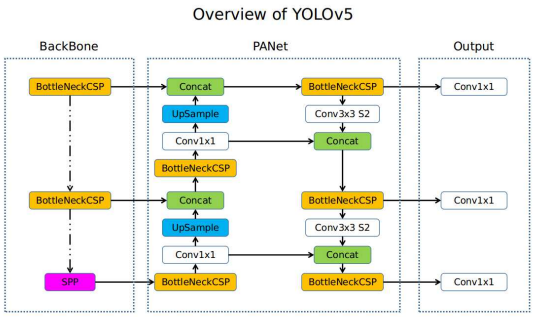

Figure 1. YOLOv5 [5]

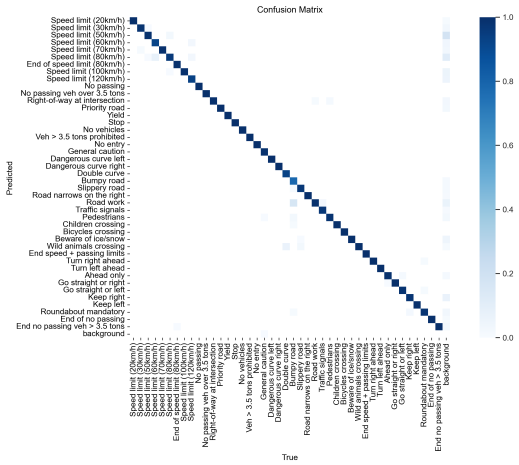

Figure 2. Confusion matrix

3. Experiments and results

3.1. Experiments

1. **Data preparation**

   (a) ***Splitting the data into train, validation, and test sets***: The original dataset was split into train/test 75/25. Applying an 80/20 train/validation split to the test dataset yielded the final 60/15/25 split.

   (b) ***Oversampling***: used the quantile-based strategy to reduce class imbalance.

   (c) ***Conversion to the required format***: YOLO uses a specific annotation format where every image is paired with an annotation file that contains descriptions of objects in it in the "class-id x-center y-center width height" format. The dataset was converted as necessary.

   (d) ***Synthesizing images***: 50% of the training images were rescaled and randomly placed within a larger background, simulating real street views.

   (e) ***Built-in YOLOv5 augmentation***: YOLOv5 applies built-in augmentations, like random changes in saturation, scale, and translations. This project used the default augmentation settings (yolov5/data/hyps/scratch-low.yaml). However, left-right flipping was disabled after the discovery that it caused confusion between the "Turn Left Ahead" - "Turn Right Ahead" and "Keep Left" - "Keep Right" class pairs.

2. **Model training**

   - ***Base model***: YOLOv5s v7.0 pre-trained on COCO val2017 [1]
   - ***Fine-tuning***: Fine-tuned using 128x128 image size (since dataset images were small), 16 batch size, and 10 epochs. All parameters were restricted by hardware constraints (NVIDIA GTX 1650 GPU (12GB VRAM)). Training took circa 5 hours.

3. **Model evaluation**
   Standard YOLOv5 evaluation script was used with the weights obtained from fine-tuning.

3.2. Results

Performance metrics across various classes can be seen in Table 1.

The confusion matrix can be seen in Figure 2.

Some examples of the model's predictions versus ground truth can be seen on Figure 3.

(a) Ground truth (b) Predictions

Figure 3. Ground truth vs predictions

4. Discussion and conclusions

4.1. Limitations and further work

- ***Dataset limitations***: while the results on the test set are impressive, the overall model is only as good as its data, and that might not be much despite my best efforts.
- ***Further evaluation*** on real street images is required to assess generalization capabilities.

Class	Images	Instances	P	R	mAP50	mAP50-95
all	12630	12630	0.993	0.981	0.991	0.89
Speed limit (20km/h)	12630	60	0.997	1	0.995	0.943
Speed limit (30km/h)	12630	720	0.996	0.977	0.991	0.909
Speed limit (50km/h)	12630	750	0.988	0.987	0.994	0.912
Speed limit (60km/h)	12630	450	0.979	0.92	0.977	0.899
Speed limit (70km/h)	12630	660	0.994	0.992	0.995	0.911
Speed limit (80km/h)	12630	630	0.954	0.917	0.976	0.876
End of speed limit (80km/h)	12630	150	0.999	1	0.995	0.82
Speed limit (100km/h)	12630	450	0.993	0.992	0.995	0.904
Speed limit (120km/h)	12630	450	0.995	0.93	0.977	0.865
No passing	12630	480	1	1	0.995	0.913
No passing veh over 3.5 tons	12630	660	0.999	0.998	0.995	0.864
Right-of-way at intersection	12630	420	0.995	0.991	0.995	0.88
Priority road	12630	690	0.998	0.997	0.995	0.911
Yield	12630	720	1	0.997	0.995	0.91
Stop	12630	270	1	1	0.995	0.902
No vehicles	12630	210	1	1	0.995	0.869
Veh over 3.5 tons prohibited	12630	150	0.999	1	0.995	0.905
No entry	12630	360	1	1	0.995	0.814
General caution	12630	390	1	0.98	0.995	0.891
Dangerous curve left	12630	60	0.998	1	0.995	0.929
Dangerous curve right	12630	90	0.999	1	0.995	0.886
Double curve	12630	90	1	0.924	0.977	0.899
Bumpy road	12630	120	1	0.775	0.939	0.875
Slippery road	12630	150	0.999	0.967	0.995	0.921
Road narrows on the right	12630	90	0.998	1	0.995	0.931
Road work	12630	480	0.969	0.99	0.994	0.936
Traffic signals	12630	180	0.992	0.967	0.989	0.88
Pedestrians	12630	60	0.971	0.967	0.994	0.959
Children crossing	12630	150	0.998	1	0.995	0.896
Bicycles crossing	12630	90	0.998	1	0.995	0.915
Beware of ice/snow	12630	150	0.991	0.993	0.995	0.881
Wild animals crossing	12630	270	0.966	1	0.991	0.908
End speed + passing limits	12630	60	0.998	1	0.995	0.83
Turn right ahead	12630	210	1	1	0.995	0.884
Turn left ahead	12630	120	0.996	1	0.995	0.894
Ahead only	12630	390	0.997	0.986	0.995	0.89
Go straight or right	12630	120	0.998	1	0.995	0.912
Go straight or left	12630	60	0.981	0.983	0.985	0.804
Keep right	12630	690	1	0.972	0.995	0.866
Keep left	12630	90	0.998	1	0.995	0.908
Roundabout mandatory	12630	90	0.99	0.967	0.988	0.928
End of no passing	12630	60	0.997	1	0.995	0.78
End no passing veh over 3.5 tons	12630	90	0.983	1	0.995	0.871

Table 1. Performance metrics for different traffic sign classes

- ***Lack of a suitable baseline***: following their common sense, which I apparently lack, other researchers only treated this dataset as a classification problem, while this project focuses on object detection.
- ***Inference speed*** should be evaluated because it is crucial for autonomous driving applications. However, the lack of baselines and hardware variations rendered any attempts on my part purposeless.

4.2. Conclusion

This project successfully adapted a classification-focused dataset for object detection, optimizing YOLOv5s for traffic sign recognition despite dataset constraints. Through data augmentation, preprocessing improvements, and model fine-tuning, strong performance was achieved on the test set (mAP@50: 99.1%, mAP@50-95: 89.0%).

However, the model's true effectiveness remains untested on street images. Future work should focus on collecting real-world data, testing generalization, and optimizing for real-time inference.

5. Contribution statement

All work for this project was done by Balashova Ekaterina, the only team member.

References

[1] G. Jocher. Yolov5 by ultralytics, 2020. https://github.com/ultralytics/yolov5. 2

[2] Yadip S. M. Mastering all yolo models from yolov1 to yolov9: Papers explained., 2024. https://learnopencv.com/mastering-all-yolo-models/. 1

[3] S. Rath and V. Gupta. Performance comparison of yolo object detection models – an intensive study., 2022. https://learnopencv.com/performance-comparison-of-yolo-models/. 1

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[5] seekFire on GitHub, 2020. https://github.com/ultralytics/yolov5/issues/280issue-650463630. 2