Traffic Sign Detection and Recognition (TSDR) Using Deep Learning

Ahmed Emad
Benha University

Mahmoud Mohamed Mahmoud
Ain Shams University

Mohamed Ashraf Abdel-Maqsoud
Benha University

Abdulrahman Mostafa Hassan
Cairo University

Abdelrahman Mostafa Abdo El-Sayed
Cairo University

Supervised by

Eng / Mahmoud Talaat

Ai Engineer at MCiT (Ministry of Communication and Information Technology)

TA at Zewail University (Artificial intelligence and Data science)

Abstract

Traffic sign detection and recognition is a critical task in the development of intelligent transportation systems, especially for autonomous vehicles. Traditional methods for detecting and recognizing traffic signs often struggle with accuracy, speed, and resource constraints, particularly in real-time applications. In this research, we address these challenges by employing deep learning techniques, specifically focusing on the YOLO (You Only Look Once) family of models and Faster R-CNN. Our objective is to develop a highly accurate yet computationally efficient model suitable for real-time traffic sign detection.

Keywords

Traffic Sign Detection
Traffic Sign Recognition
Deep Learning
YOLOv10m
Model Pruning
Real-time Object Detection
Autonomous Vehicles

Introduction

Object detection is an essential component in computer vision applications, ranging from autonomous vehicles to security systems. In recent years, models like Faster R-CNN and YOLO (You Only Look Once) have gained prominence for their balance between accuracy and computational efficiency. This paper presents a comparative analysis of several object detection models, including Faster R-CNN, YOLOv5 variants, and a pruned YOLOv10m model, evaluating their performance in terms of precision, recall, speed, and resource consumption. The impact of pruning on model efficiency is also explored.

Methods

Model Selection

Several object detection models were considered for this study:

- Faster R-CNN: A popular, accurate but slower model with 41,139,286 parameters.

- YOLOv5n: A lightweight YOLO variant with 2,712,500 parameters, designed for speed.

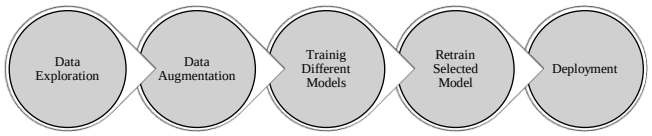
- YOLOv5s: A slightly larger model (8,077,188 parameters) offering a balance between speed and accuracy.

- YOLOv10m: A mid-range YOLO variant with 16,500,340 parameters, chosen for further pruning experiments.

- YOLOv8l: A large YOLO model with 44,275,338 parameters.

- Roboflow 3.0 Object Detection: Included for comparison with more recent advancements in object detection technologies.

Methodology



Pruning Approach

The YOLOv10m model was pruned using a 5% pruning strategy, targeting connections with low weight magnitudes. This approach aimed to reduce the number of parameters and the computational complexity while maintaining high accuracy. The pruned model, with 16,500,340 parameters, was evaluated against the original to assess any degradation in performance.

Experiments

Dataset

The models were trained and tested on a large-scale dataset containing various object categories and scenarios typical of real-world environments. The dataset was split into training and validation sets to ensure proper evaluation of model performance.

Evaluation Metrics

Performance was evaluated using the following metrics:

- **Precision (%):** The ability of the model to correctly identify objects.
- **Recall (%):** The ability to detect all relevant objects in an image.
- **F1 Score (%):** The harmonic mean of precision and recall.
- **Mean Average Precision (mAP) (%):** A standard metric for object detection.
- **Speed (ms):** Inference speed, measured in milliseconds per image.
- **Size (MB):** The size of the model, a key consideration for deployment in resource-constrained environments.
- **GFLOPs:** Number of floating-point operations per second, indicating computational cost.

Pruning Impact

After pruning 5% of the YOLOv10m model's weights, the performance impact was minimal. Precision dropped slightly from 95.95% to 94.29%, while the recall remained nearly unchanged. The size and inference speed were maintained, demonstrating that pruning effectively reduces model complexity without significantly sacrificing performance.

Discussion

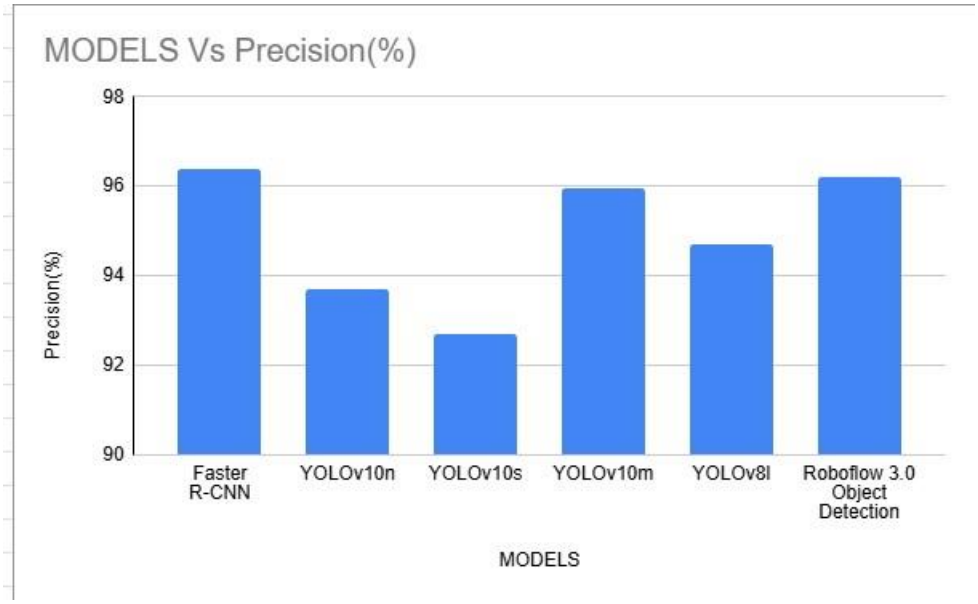
This analysis shows that Faster R-CNN, while highly accurate, comes with a significant computational burden, making it unsuitable for real-time applications. The YOLO family of models, particularly YOLOv10m, offers a favorable balance between precision and speed. Pruning YOLOv10m by 5% demonstrates that model compression techniques can effectively reduce complexity with minimal impact on performance, providing a viable solution for deploying models in environments with limited computational resources.

The pruned YOLOv10m maintained comparable precision and recall to its unpruned counterpart, confirming the effectiveness of the pruning technique used. The speed and size of the model remained largely unchanged, indicating that this approach can be valuable in optimizing model efficiency.

Results

The table below summarizes the performance of each model:

Model	Parameters	Precision (%)	Recall (%)	Size (MB)	Speed (ms)	GFLOPs	F1 Score (%)	mAP
Faster R-CNN	41,139,286	96.4	93.8	330.6	102.02	134.4	95.1	95.1
YOLOv10n	2,712,500	93.7	85.3	5.7	1.149	8.421	89.3	89.3
YOLOv10s	8,077,188	92.7	90.3	16.5	2.446	24.826	91.48	91.48
YOLOv10m	16,500,340	95.95	89.5	33.5	5.25	64.051	92.6	92.6
YOLOv8l	44,275,338	94.7	95.0	83.6	18.41	107.925	94.85	94.85



Pruning Impact

After pruning 5% of the YOLOv10m model's weights, the performance impact was minimal. Precision dropped slightly from 95.95% to 94.29%, while the recall remained nearly unchanged. The size and inference speed were maintained, demonstrating that pruning effectively reduces model complexity without significantly sacrificing performance.

Discussion

This analysis shows that Faster R-CNN, while highly accurate, comes with a significant computational burden, making it unsuitable for real-time applications. The YOLO family of models, particularly YOLOv10m, offers a favorable balance between precision and speed. Pruning YOLOv10m by 5% demonstrates that model compression techniques can effectively reduce complexity with minimal impact on performance, providing a viable solution for deploying models in environments with limited computational resources.

The pruned YOLOv10m maintained comparable precision and recall to its unpruned counterpart,

confirming the effectiveness of the pruning technique used. The speed and size of the model remained largely unchanged, indicating that this approach can be valuable in optimizing model efficiency.

MODELS	Precision(%)	Recal(%)	mAP
YOLOv10m	95.95	89.5	0.841
YOLOv10m_pruned5%	94.78	89.674	0.827
YOLOv10m_pruned20%	68.079	60.932	0.523

Conclusion

The study demonstrates the advantages of YOLO models in terms of speed and resource consumption, with YOLOv10m emerging as a strong candidate for real-time applications. Pruning further enhances the model's efficiency without significantly impacting accuracy, making it suitable for deployment in constrained environments. Further research could explore more aggressive pruning strategies or apply the technique to other YOLO variants for greater optimization.

Tools

- TensorFlow
- YOLO (You only Look Once)
- Albumentations
- Google Colab
- Numpy
- Docker
- Microsoft Auzre
- Kaggle Notebook
- Python
- Visual studio code

REFERENCES

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016).
You Only Look Once: Unified, Real-Time Object Detection.
In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[2] **Bochkovskiy, A., Wang, C., & Liao, H. M. (2020).**
YOLOv4: Optimal Speed and Accuracy of Object Detection. This paper introduces YOLOv4, a more efficient and accurate version of YOLO, providing state-of-the-art results in object detection tasks.

[3] **Zhu, P., Wen, L., Du, D., Bian, X., Hu, Q., & Liu, J. (2021).**
Detection and Tracking Meet Drones Challenge: VisDrone 2021.
In Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops.

[4] "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron

***Description**: This book provides practical examples and code for implementing machine learning and deep learning models using popular frameworks like TensorFlow. It includes chapters on object detection, neural networks, and practical implementations of CNNs.*