License Plate Recognition: Comparative Analysis of Object Detection Models

Author: Eric Jonas - Course: Learning from Images (WiSe24)

Automatic License Plate Recognition (ALPR) is crucial for various applications like traffic monitoring, automated toll collection and parking management. This project explores various object detection algorithms, evaluates their performance and demonstrates their effectiveness through a prototype ALPR system.

Problem

- ALPR systems must detect, track and extract license plates accurately from images and videos.
- Rather than detecting entire vehicles, our system focuses directly on license plates. This approach can enhance performance, but also presents challenges, such as making object tracking more difficult due to the smaller bounding boxes.
- The goal is to compare various object detection methods using key metrics such as precision, recall, and inference speed.

Related Work

- Several ALPR solutions already exist, such as OpenALPR, along with numerous research papers that focus on specific aspects of ALPR systems, like optimizing the OCR process and improving tracking methods. [1]
- Object detection methods range from classical edge detection to modern deep learning models. State-of-the-art approachinclude YOLO and Detection Transformers (DETR), which offer high accuracy and efficiency. [2]

Dataset & Preprocessing

Large Dataset: ~26,000 images from (YOLO format annotations). [3]
Small Dataset: ~400 high-quality images (VOC format, converted to YOLO). [4]
Merged Dataset: Combined into datasets for training, validation and testing.

Pascal VOC Format

Stores image metadata in XML format, along with object class labels and bounding box coordinates (xmin, ymin, xmax, ymax).

YOLO Format

Stores image annotations in a text file, with object class labels and bounding box coordinates normalized relative to image width and height (x_center, y_center, width, height).

Augmentations: Random rotations, snow effects and contrast adjustments to increase diversity and model robustness.

Implementation

All models were fine-tuned from pretrained COCO weights.

Faster R-CNN (Two-stage Detector):
We implemented two Faster R-CNN models using different backbones: ResNet50 and MobileNet. ResNet50 offers higher accuracy but comes at the cost of slower inference, while MobileNet delivers faster performance with reduced accuracy. Although Faster R-CNNs achieve high detection accuracy, their slower inference times make them less suitable for real-time applications. The implementation was done with PyTorch. [5]

YOLO (One-stage detector):
YOLO is designed for fast, real-time performance, making it ideal for applications requiring quick detection. However, it sacrifices some accuracy for speed. The model is implemented using the Ultralytics library, which simplifies training various YOLO models. [6]

DETR (Transformer-based detection):
DETR uses a transformer-based, end-to-end detection model with self-attention mechanisms for object detection. We attempted to implement DETR using PyTorch, but due to slower inference times and an issue we encountered during training, we decided to discontinue this approach. [7]

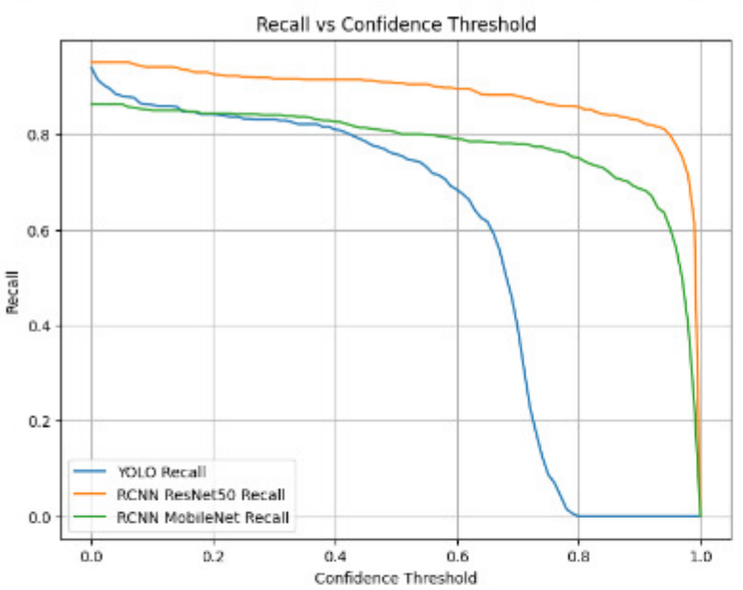
Object Tracking with DeepSORT:
DeepSORT tracks objects across frames, predicts their movement, and associates new bounding boxes with existing ones. This object tracking enables us to link objects together, making it easier to detect inconsistencies in OCR predictions.

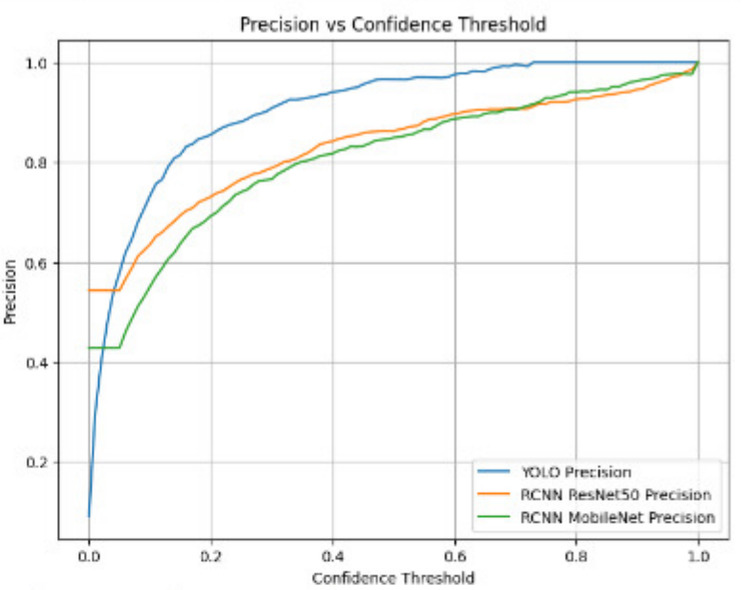
OCR (Text Extraction):
We used EasyOCR to extract text from detected license plates, but the results were inconsistent due to poor lighting and distortions. Future improvements could include fine-tuning OCR models specifically for license plates to improve prediction accuracy.

Results

Models were evaluated with a **minimum confidence threshold of 0.25** and a **minimum IoU of 0.5**. Note that the total number of detections (TP + FP + FN) differs among models because low-confidence predictions are discarded.

Model	Precision	Recall	TP	FP	FN	mAP50	mAP50-95	Training Duration (Epochs)
Faster R-CNN ResNet50	0.7655	0.9200	483	148	42	0.9706	0.7792	3 (~6h)
Faster R-CNN MobileNetv3 (small)	0.7404	0.8419	442	155	83	0.9549	0.7749	3 (~3h)
YOLOv11 (n)	0.8810	0.8324	437	59	88	0.9354	0.8001	10 (~2h)





Green: Ground Truth - Blue: Predicted with Confidence Value

Faster R-CNN ResNet50

Faster R-CNN MobileNetv3

YOLOv11

Overall, YOLO tends to predict many bounding boxes, which are filtered out due to low confidence, leaving a high proportion of accurate predictions. In contrast, the Faster R-CNN models generate less bounding boxes but with higher confidence and a greater likelihood of being correct.

Prototype

Finally, we developed a prototype that mimics the functionality of an ANPR system. The prototype consists of the following components:

1. Frame-by-frame Analysis: We Predict bounding boxes around license plates in each frame and feed these into the tracking algorithm.

2. Tracking: The license plates are tracked across frames. Configuration settings include max_iou_difference 0.99 and max_age 10.

3. OCR: The text from the license plates is extracted and compared to the previously recognized plate text.

Video

https://youtu.be/h7skTEX-wGM

Average Processing Time per Frame

Resnet50 Faster R-CNN: too slow
MobileNetv3 Faster R-CNN: 0.21s
YOLOv11: 0.13s

Conclusion

Faster R-CNN with a ResNet50 backbone achieved the highest recall and mAP50 scores, making it the best for accurate license plate detection. Yolov11 offered a strong balance of precision, speed and recall, making it ideal for real-time applications. MobileNetV3 was efficient but performed worse than both ResNet50 and YOLOv11. For object tracking, DeepSORT reliably maintained license plate identification across frames. However, OCR performance was challenging, with easyOCR struggling to consistently predict the same text across frames.

References

1. Papers with Code. (n.d.). Object detection on COCO. Papers with Code. Retrieved from https://paperswithcode.com/sota/object-detection-on-coco
2. Sun, Y., Sun, Z., & Chen, W. (2024). The evolution of object detection methods. Computers in Industry, 156, 107078.
3. Elmenshawii, F. (2024). Large-License-Plate-Detection-Dataset. Kaggle. Retrieved from https://www.kaggle.com/datasets/fareselmenshawii/large-license-plate-datasetlarge-license-plate-dataset/code
4. Larxel. (2020). *Car license plate detection* Kaggle. Retrieved from https://www.kaggle.com/datasets/andrewmvd/car-plate-detection
5. Cai, X. (2021, April 1). Understanding Faster R-CNN from the perspective of implementation. Retrieved from https://www.lablab.top/post/how-does-faster-r-cnn-work-part-i/
6. Hugging Face. (n.d.). DETR. Retrieved from https://huggingface.co/docs/transformers/model_doc/detr
7. Kundu, R. (2023). YOLO: Algorithm for object detection explained. V7 Labs. Retrieved from https://www.v7labs.com/blog/yolo-object-detection