CSE 573 - Computer Vision and Image Processing

University at Buffalo
Department of Computer Science and Engineering

Devendra Mandava, dmandava, 50534296

Final Report: Vision-Based Automatic License Plate Recognition for Road Safety

🗃️ Table of Contents

📰 Overview of the Project

Application

The License Plate Detection System is designed to automatically detect vehicles, locate license plates, and extract textual information from plates in real-time. This application has significant real-world implications, including traffic law enforcement, toll collection, parking management, and vehicle tracking. By processing images or videos, the system aims to provide accurate and efficient license plate recognition, even under challenging conditions such as varying lighting or motion blur.

State of the Art

Current state-of-the-art methods for object detection include models like Faster R-CNN and YOLO. Faster R-CNN offers high accuracy but often at the cost of slower inference times, making it less suitable for real-time applications. YOLO (You Only Look Once), on the other hand, balances accuracy and speed, making it an ideal candidate for tasks requiring real-time performance. For Optical Character Recognition (OCR), tools like EasyOCR are widely used due to their versatility and robustness in handling various fonts and languages.

In this project, both Faster R-CNN and YOLOv8 were tested for vehicle and license plate detection. YOLOv8 significantly outperformed Faster R-CNN in terms of inference speed and suitability for real-time applications without a notable compromise in detection accuracy. As a result, YOLOv8 was selected as the primary model for this system.

Contributions

This project presents the following contributions:

1. **Custom Fine-Tuning of YOLOv8**: A pre-trained YOLOv8 model was fine-tuned on a custom dataset specifically annotated for license plate detection.
2. **OCR Integration with Preprocessing**: An OCR pipeline using EasyOCR was developed, incorporating preprocessing techniques such as resizing and thresholding to improve text recognition accuracy.
3. **Vehicle Tracking with SORT**: The SORT (Simple Online and Realtime Tracker) algorithm was integrated to maintain vehicle identities across video frames.
4. **Pipeline Development**: A seamless pipeline combining vehicle detection, license plate extraction, and OCR was implemented for both image and video inputs.
5. **Web Interface**: A Flask-based web application was created to allow users to upload images or videos and visualize detection results in real-time.

▼ 🚀 Approach

🤖 Algorithms Used

1. **YOLOv8 for Object Detection**:
   YOLOv8 was utilized for detecting both vehicles and license plates in

images and videos. The model was fine-tuned on a custom dataset to enhance detection accuracy for license plates in diverse conditions, including variations in lighting and motion blur. Its real-time processing capability made it ideal for this application.

2. **OCR with EasyOCR**:
   The EasyOCR library was employed to extract textual information from the detected license plates. Preprocessing steps such as resizing and thresholding were implemented to improve recognition performance. These steps ensured better accuracy, even for plates with poor image quality or non-standard fonts.

3. **Faster R-CNN for Initial Testing**:
   Faster R-CNN was evaluated during the initial phase of the project for object detection. However, due to its slower inference time, it was deemed unsuitable for real-time applications, leading to the adoption of YOLOv8.

4. **SORT for Tracking**:
   The SORT algorithm was integrated to track vehicles across multiple frames in video inputs. This ensured continuity and identity preservation for detected objects, providing a robust solution for dynamic scenes.

💻 Self Coded Components

1. **YOLOv8 Training Pipeline**:
   A custom training pipeline was developed to fine-tune YOLOv8 on a dataset annotated specifically for license plate detection. The dataset included bounding box labels for vehicles and license plates.

2. **OCR Preprocessing**:
   The preprocessing module was coded to enhance the OCR pipeline. It included resizing license plate regions to a consistent size, applying grayscale conversion, and thresholding techniques to filter noise and improve text clarity.

3. **Pipeline Integration**:
   The overall pipeline was designed to seamlessly combine detection, tracking, and OCR. This integration enabled efficient processing of both image and video inputs, with results visualized through bounding boxes and extracted text.

🌐 Online Resources

1. **EasyOCR Library**:
   EasyOCR, an open-source OCR library, was used for text extraction. Its robust performance in recognizing alphanumeric characters made it a valuable addition to the pipeline.
   https://github.com/JaidedAI/EasyOCR
2. **SORT Tracker Implementation**:
   The SORT tracking algorithm was integrated from its publicly available implementation, with minor modifications to suit the project's requirements.
   https://github.com/abewley/sort

▼ 🔍 Experimental Protocol

📊 Datasets Used

The **CCPR dataset** was used as the primary source for training and testing the system. This dataset includes a diverse collection of vehicle images with annotated license plates, capturing variations in lighting, angles, and plate designs. Its relevance lies in providing realistic challenges that align closely with real-world scenarios, ensuring that the model generalizes well.

📈 Evaluation Metrics

Success was evaluated using both qualitative and quantitative measures:

- **Qualitative Evaluation**: Visual inspections of the output images and videos, ensuring bounding boxes align with vehicles and license plates accurately, and extracted text matches the visible plate information.
- **Quantitative Evaluation**: Metrics such as Precision, Recall, and F1-Score were calculated for license plate detection. OCR accuracy was measured as the percentage of correctly identified plate texts.

💻 Compute Resources

The project was developed and tested using:

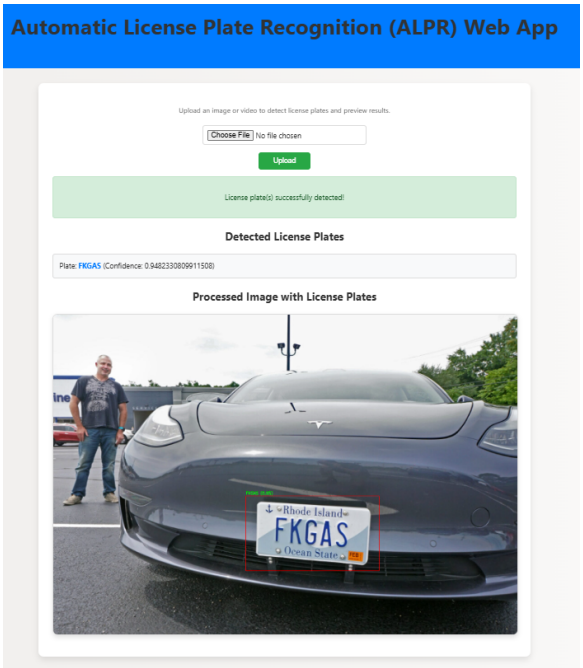
- **Hardware**: Google Colab with access to NVIDIA GPUs for model training, and a local machine with an Intel Core i7 processor for inference testing.

- **Software**: Python-based libraries including OpenCV, PyTorch, and EasyOCR. Google Colab provided a cloud-based environment for efficient experimentation.

▼ 💭Results

Qualitative Results

Qualitative results are demonstrated using annotated images and videos showing bounding boxes around detected vehicles and license plates. OCR results are overlaid on the images, displaying the extracted text along with confidence scores.



Comparison with State-of-the-Art

The performance of YOLOv8 for license plate detection was compared with Faster R-CNN, which was initially tested. YOLOv8 showed significantly faster inference times, making it more suitable for real-time applications. While Faster R-CNN achieved slightly higher detection precision on static images, YOLOv8 outperformed in terms of detection speed and robustness in dynamic scenarios.

Interpretation and Presentation of Results

The results indicate that:

1. The model effectively detects license plates under varying lighting and motion conditions.
2. OCR accuracy is high for standard license plate fonts but slightly decreases for non-standard or degraded plates.
3. Tracking using SORT ensures consistent identity assignment across video frames, reducing redundant detections.

Conclusions from Results

The integration of YOLOv8, EasyOCR, and SORT demonstrates a well-balanced approach to license plate recognition. The system achieves real-time performance without significant compromises in accuracy, making it suitable for practical applications such as traffic monitoring and toll management. The adaptability of the pipeline ensures its effectiveness across diverse scenarios.

▼ 🛠️ Analysis

Advantages

1. **Real-Time Performance**: The use of YOLOv8 enables fast inference, allowing the system to process videos in real time.
2. **Robustness**: The system performs well under varying conditions, including different lighting scenarios, motion blur, and diverse plate designs.
3. **Scalability**: The integration of SORT for tracking ensures the system can handle multiple vehicles effectively without redundant detections.
4. **Generalization**: Fine-tuning YOLOv8 on a custom dataset ensures that the model generalizes well across various environments and plate designs.

Limitations

1. **OCR Challenges**: Non-standard fonts, poor resolution, or occlusions in license plates occasionally lead to incorrect or missed text extraction.

2. **Dependence on Dataset Quality**: The model's performance is closely tied to the quality and diversity of the dataset used for training. Any gaps in dataset representation may lead to reduced effectiveness in unseen scenarios.
3. **Tracking Accuracy**: While SORT is effective for simple tracking, it can struggle in highly crowded scenes or when vehicles overlap significantly.
4. **Hardware Dependence**: Real-time performance depends on the availability of GPUs, which may not always be feasible in low-resource environments.

Presentation of Analysis

The balance between accuracy and speed achieved by YOLOv8, combined with EasyOCR and SORT, demonstrates a strong use case for real-world applications. However, addressing OCR limitations and improving dataset diversity could further enhance the system's reliability. Future improvements, such as adopting more advanced tracking algorithms or enhancing preprocessing for OCR, could mitigate some of these limitations and expand the scope of the application.

▼ 👩‍💻Discussion and Lessons Learned

Lessons Learned

1. **Integration of Multiple Components**: Successfully integrating object detection, tracking, and OCR into a cohesive pipeline highlighted the importance of modular design and clear interfaces between components.
2. **Model Fine-Tuning**: Fine-tuning YOLOv8 on a custom dataset demonstrated the value of adapting pre-trained models to specific use cases, leading to significant improvements in detection accuracy.
3. **Real-Time Processing Challenges**: Addressing the challenges of real-time processing, such as optimizing model inference speed and ensuring low latency, provided insights into balancing accuracy and performance.
4. **Data Preprocessing for OCR**: Preprocessing steps, such as resizing and thresholding, were critical for improving OCR accuracy, emphasizing the importance of tailored data preparation.

Future Applications

The knowledge gained from this project can be applied to:

1. Developing advanced traffic monitoring systems that integrate license plate recognition with vehicle behavior analysis.
2. Expanding the system for use in toll booths or parking management systems where automation is key.
3. Enhancing the pipeline with deep learning-based tracking algorithms for more robust performance in complex scenarios.

This project provided a valuable learning experience in combining multiple technologies to solve a real-world problem, laying the foundation for further research and development in computer vision applications.

💭Bibliography

1. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. Retrieved from https://arxiv.org/abs/1804.02767
2. Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple Online and Realtime Tracking. 2016 IEEE International Conference on Image Processing (ICIP). https://doi.org/10.1109/ICIP.2016.7533003
3. Baek, J., Kim, G., Lee, J., Park, S., & Kim, H. (2019). Character Region Awareness for Text Detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/CVPR.2019.00649