Smart Parking Enforcement System Using License Plate Recognition

A next-generation approach to campus parking management using computer vision technology to create a more efficient, fair, and user-friendly parking experience.

Team 212

The Problem & Research

Current Challenges

Parking violations often go unnoticed in real-time. Manual enforcement is error-prone and costly.

ASU's current system relies on manual patrols and basic scanning devices.

Research Findings

Computer Vision offers automation for vehicle tracking and payment validation.

Manual systems struggle during peak hours. Real-time data remains largely underutilized.

Proposed Solution

Smart Parking Enforcement using License Plate Recognition to automate the enforcement process.

This system will improve efficiency and user satisfaction across campus.

Why This Matters

Reduces Errors

Prevents unfair ticketing through accurate license plate identification.

Optimizes Resources

Enforcement staff can focus on genuine violations and customer service.

Supports Planning

Provides valuable data for infrastructure development and lot utilization.

Key Stakeholders

Benefits Parking Services, students, faculty, staff, visitors, and campus security.

End-to-End Solution Overview

1

Image Capture

High-definition cameras installed throughout parking areas capture vehicle plates.

2

Plate Detection

Computer vision model identifies and isolates license plates in captured images.

3

Database Cross-Check

System verifies plate against parking payment database instantly.

4

Dashboard Updates

Operations center receives real-time violation and occupancy data.

5

Digital Ticketing

System issues electronic citations for confirmed violations.

Scope & Success Metrics

Pilot Scope

- ASU Tempe Campus initial deployment
- Focus on high-traffic parking structures
- 3-month evaluation period

Success Metrics

- 90%+ plate detection accuracy
- 20% reduction in ticket complaints
- 25% reduction in enforcement costs

Resources Required

- HD cameras at entry/exit points
- GPU-enabled environment (for model training and fine-tuning)
- Cloud-based monitoring dashboard

Risk Mitigation

Recognition Challenges

Poor lighting and damaged plates may reduce accuracy.

Technical Solutions

Model tuning and synthetic data augmentation improve performance.

Privacy Concerns

License plate data requires proper security protocols.

Data Protection

Encryption and limited retention periods safeguard personal information.

CV Model Workflow

Image Capture

HD webcams strategically positioned capture vehicle images.

Plate Detection

YOLOv8 algorithm identifies license plate location within the image.

OCR Recognition

TrOCR extracts the alphanumeric plate number from the detected region.

Fee Calculation

System checks payment status and applies appropriate fee structure.

Model Details

Training Configuration

- **Base Model**: YOLOv8s (pretrained on COCO dataset)
- **Fine-tuning Approach**:
  - Transfer learning on custom car plate dataset
  - Updated weights to specialize in plate detection
- **Dataset**:
  - Custom annotated images (specified in `car_plate.yaml`)
  - Train-validation split handled by YOLO automatically
- **Epochs**: 50
- **Image Size**: 640x640 pixels
- **Batch Size**: 16
- **Optimizer**: AdamW
- **Loss Functions**:
    - Distribution focal loss
    - Objectness loss
    - Classification loss

Environment Setup for text detection

- **TrOCR**: OCR model for text extraction from plates
- **Config**:
  - **YOLO Weights**: Path to trained model
  - **Video Source**: Webcam or video file
  - **Confidence Threshold**: 0.2
  - **Padding for OCR**: 20px
  - **Beam Size (TrOCR)**: 5
- **Optimization**:
  - **TrOCR**: FP16 on GPU for speed

Solution Notebook: https://colab.research.google.com/drive/17ddclCH5_ZFycnwfvKGrql_XE8kKLufh?usp=sharing

Technical Implementation Details

Validation Results

mAP50: 0.98, mAP50-95: 0.74

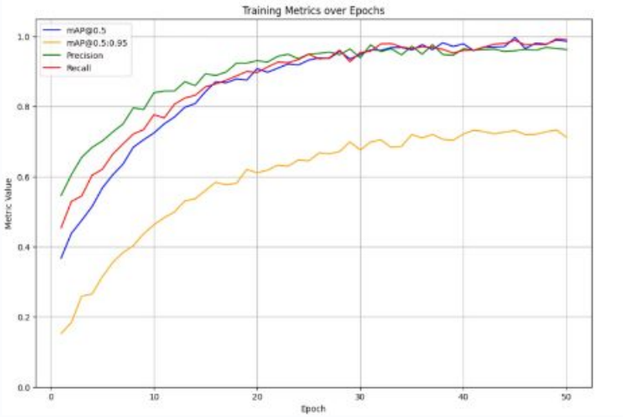
Performance Metrics

Precision: 0.96, Recall: 0.98

Training Data

Kaggle dataset

Training metrics over epochs




```
Validating runs/detect/plate_detector_20ep/weights/best.pt...
Ultralytics 8.3.118 🚀 Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
Model summary (fused): 72 layers, 11,125,971 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 14/14 [00:02<00:00,  6.46it/s]
                   all        433        471      0.963      0.986      0.988      0.739
Speed: 0.1ms preprocess, 0.9ms inference, 0.0ms loss, 1.0ms postprocess per image
```

Demonstration

5A22845 processed ➜ Outsider ➜ Pay $10

	Name	ASU ID	Car Number	Amount to Pay	Timestamp
0	Outsider	-	5A22845	$10	2025-04-27 18:34:02

Conclusion

1

High Detection Accuracy

The fine-tuned YOLOv8 model achieves high precision in detecting license plates from video feeds in real-time.

2

Effective OCR

The TrOCR text recognition model accurately extracts alphanumeric text from detected plates.

3

Seamless Integration

The end-to-end pipeline combines YOLO for detection and TrOCR for OCR, providing a robust license plate recognition system.

Findings

- High detection accuracy with adjustable confidence threshold
- TrOCR effective for plate text extraction, but struggles with low contrast or blurry images
- Real-time integration with webcam enables practical use for security and vehicle tracking

Limitations

- Accuracy drops in low-light or low-resolution conditions
- Difficulty detecting multiple plates in a single frame
- Requires GPU for optimal real-time performance

Future Enhancements

1 **Improve Detection Accuracy**

Fine-tune YOLO and OCR models for higher precision

2 **Multi-Plate Handling**

Detect and process multiple license plates per image

3 **Payment Integration**

Integrate with PayPal or Stripe for seamless payments

4 **Low-Light Performance**

Enhance model accuracy for night-time or poor lighting conditions

Task ownership

Problem and Dataset Selection	Kartik, Kriti, Minsoo, Ravi, Vindhya
Solution Ideation	Kartik, Kriti, Minsoo, Ravi, Vindhya
Model Training and Testing	Minsoo, Ravi
Presentation Preparation	Kartik, Kriti, Minsoo, Ravi, Vindhya

References

- **Kaggle License Plate Data**. (n.d.). *Car Plate Detection Dataset*. Retrieved from https://www.kaggle.com/datasets/andrewmvd/car-plate-detection/data
- **Ultralytics**. (2024). *YOLOv8 – Object Detection*. Retrieved from https://docs.ultralytics.com/
- **Microsoft Research**. (n.d.). *TrOCR – OCR Model*. https://github.com/huggingface/huggingface_hub
- **OpenCV**. (n.d.). *OpenCV – Image Processing*. Retrieved from https://opencv.org/
- **PyTorch**. (n.d.). *PyTorch – Deep Learning Framework*. Retrieved from https://pytorch.org/

Thank You!