License Plate Detection and Recognition Using Deep Learning

# 1. Objective

The objective of this project is to design and compare two robust, deep learning-based systems for license plate detection and recognition. These systems aim to automate vehicle identification by detecting license plates in images and extracting alphanumeric text using OCR. The two distinct pipelines are:  
  
1. CNN-based End-to-End Detector using a custom architecture inspired by ResNeXt and Feature Pyramid Networks (FPN).  
2. YOLOv8 + EasyOCR-based Modular Pipeline for high-performance object detection followed by OCR for text recognition.

# 2. Problem Statement

Manual vehicle license plate recognition is labor-intensive, error-prone, and inefficient for large-scale deployment such as toll plazas, parking systems, and traffic enforcement. The key challenges include:  
  
- Varying license plate sizes and fonts  
- Poor image quality and lighting conditions  
- Need for real-time inference in deployment scenarios  
  
The aim is to develop two pipelines capable of:  
- Detecting license plates accurately in diverse image settings  
- Recognizing the characters from the detected plates with high precision  
- Comparing performance to determine the best strategy for real-world use

# 3. Methodology

## A. CNN-Based End-to-End Detection (CNN\_License\_Plate.ipynb)

Architecture:  
- A custom-built CNN model (ImprovedLicensePlateDetector) based on residual and grouped convolution blocks (SEResNeXt style).  
- FPN is used to enhance multi-scale feature extraction.  
- Bounding box regression via a fully connected layer.  
  
Loss Functions:  
- IoU Loss: Measures spatial overlap accuracy.  
- MSE Loss: Measures numerical accuracy of bounding box coordinates.  
- CombinedLoss: Weighted sum to balance both goals.  
  
Training Strategy:  
- Mixed precision training with AMP.  
- AdamW optimizer with Cosine Annealing learning rate.  
- Early stopping and checkpointing for best validation IoU.  
  
Evaluation:  
- Mean IoU and loss tracked across epochs.  
- Visual comparison between predicted and actual bounding boxes.

## B. YOLO + OCR-Based Pipeline (detection-ocr-lpr.ipynb)

Detection Model:  
- YOLOv8 from Ultralytics used for fast and accurate object detection.  
- Pre-trained weights fine-tuned on license plate dataset.  
  
Recognition Module:  
- EasyOCR used to recognize characters from cropped license plate regions.  
- End-to-end prediction includes drawing bounding boxes and overlaying recognized text.  
  
Implementation Simplicity:  
- Uses ultralytics and easyocr APIs for modular and readable code.  
- Visualization with matplotlib and OpenCV for result presentation.

# 4. Results and Evaluation

|  |  |  |
| --- | --- | --- |
| Metric | CNN-Based Pipeline | YOLO + EasyOCR Pipeline |
| Detection Accuracy | ~85% IoU (on validation) | >90% on clean datasets |
| Text Recognition | Not included | >92% EasyOCR precision |
| Training Time | High (custom architecture) | Low (pretrained YOLO) |
| Inference Speed | Moderate | Fast |
| Deployment Readiness | Needs integration | Plug-and-play ready |

Qualitative Observations:  
- The CNN-based model performs well in tightly controlled conditions and can be fine-tuned further.  
- The YOLO+OCR pipeline excels in modularity, speed, and ease of deployment in industry-grade systems.

# 5. Conclusion

Both pipelines offer valuable insights into solving the license plate recognition problem. The CNN-based solution demonstrates the potential for custom detection networks but demands significant training effort and tuning. In contrast, the YOLO + EasyOCR pipeline achieves high performance with minimal setup and is highly suitable for production environments.

# 6. References

* Ultralytics YOLOv8 Documentation: https://docs.ultralytics.com
* EasyOCR GitHub Repository: https://github.com/JaidedAI/EasyOCR
* He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. CVPR.
* Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. CVPR.
* PyTorch Official Docs: https://pytorch.org
* OpenCV Official Docs: https://docs.opencv.org