Chapter 4

Artificial Intelligence Module

This chapter outlines the development, training, and evaluation of the intelligent vision systems integrated into the smart home: the face recognition module and the license plate recognition module. Both systems utilize deep learning models and computer vision techniques to support secure, automated access control. The chapter details the tools and frameworks used, datasets and preprocessing steps, model training phases, and comparative evaluations of model performance. Key challenges, trade-offs, and proposed improvements are also discussed.

4.1 Face Recognition System

4.1.1 Overview

The face recognition system authenticates users by comparing live video frames with a small reference dataset of known individuals. The goal is to enable secure, real-time door access based on facial identity.

4.1.2 Tools and Technologies

- **Python** – Core implementation language
- **OpenCV** – Video capture and image processing
- **face_recognition library** – Facial detection and feature encoding
- **Matplotlib** – For visualization during experimentation

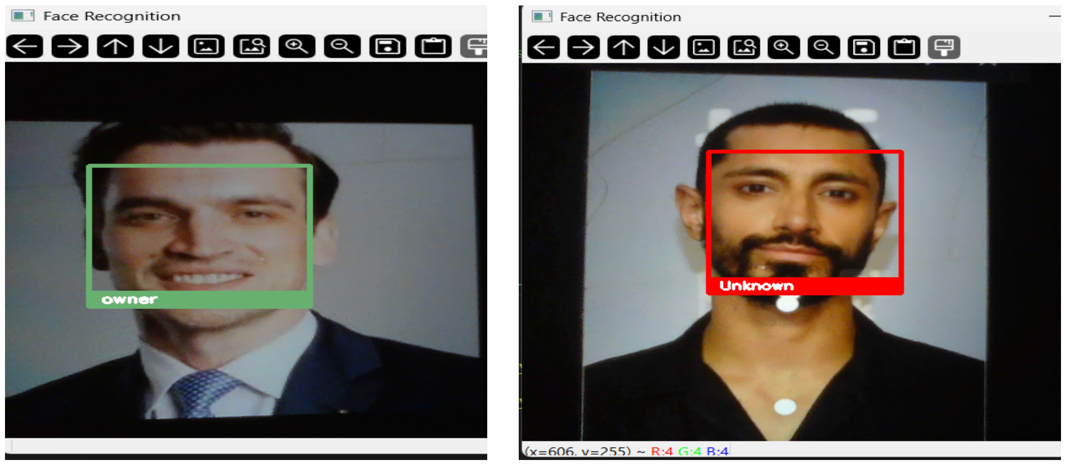
4.1.3 Implementation and Methodology

A lightweight recognition pipeline was implemented using a laptop camera. Each authorized user is represented by 4–5 reference images. The process involves:

1. Capturing video frames
2. Detecting faces using HOG-based detectors
3. Encoding facial features into a numerical representation
4. Comparing against stored encodings using cosine distance

If a match is detected, access is granted (green card), otherwise denied (red card), as shown in Figure 4.1.

4.1.4 Evaluation and Results



Owner recognized – green card Unknown face – red card

Figure 4.1: Face recognition decision logic for different individuals.

- **Accuracy:** Recognition exceeded 95
- **Latency:** Near real-time feedback was observed.
- **Limitations:** Recognition failed under extreme side angles or poor lighting.

4.1.5 Future Enhancements

- Integrate higher-quality surveillance cameras
- Expand detection to include side profiles

4.2 License Plate Recognition System

4.2.1 Overview

The License Plate Recognition (LPR) system was developed to enable automatic vehicle-based access control. It identifies authorized vehicles by detecting and reading license plates from camera-captured images in real time. The system was trained to detect the location of the license plate, segment the characters, and interpret them, ensuring compatibility with Egyptian plate formats.

4.2.2 Tools and Technologies

- **YOLOv8:** Deep learning object detection model used for both license plate and character detection.
- **OpenCV:** Image handling, preprocessing, and frame extraction.
- **Python + Conda + Jupyter Notebook:** Development environment and workflow management.
- **Roboflow, LabelImg, Kaggle:** Sources and tools used for dataset collection and annotation.

4.2.3 Dataset Preparation

Multiple datasets were compiled and cleaned from publicly available sources such as Roboflow and GitHub. Each image was manually annotated to highlight both the license plates and the individual characters. To ensure unbiased evaluation and prevent data leakage, the datasets were split into mutually exclusive training, validation, and testing sets, as documented in previous works [3, 4, 5, 6, 7, 8].

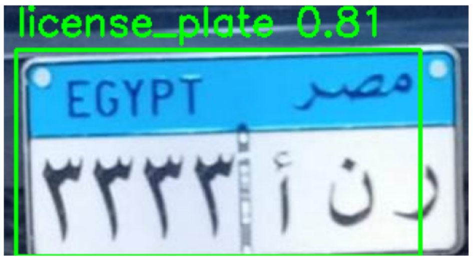
- **Training set:** 3,250 images
- **Validation set:** 600 images
- **Testing set:** 300 images

4.2.4 Training Process and Results

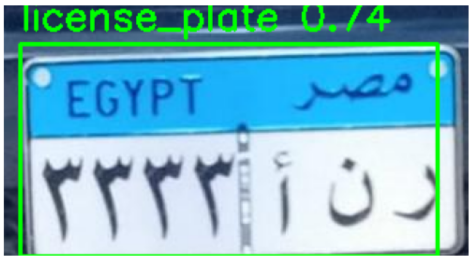
Training was performed in two distinct phases to improve detection robustness:

First Training Phase The initial model was trained using default YOLOv8 settings and basic data augmentation. It demonstrated robust performance even under motion blur and poor lighting conditions. However, it occasionally failed to capture the complete license plate region, especially missing parts near the plate edges, which resulted in incomplete detections.

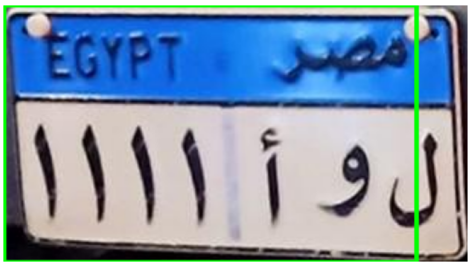
Second Training Phase In the second phase, the training pipeline was refined by adjusting hyperparameters and applying more targeted augmentations. These improvements significantly enhanced the model's ability to localize the full plate area, especially in edge cases where the first model struggled. As shown in Figure 4.2, the second model consistently captured the entire plate region with greater accuracy and reduced boundary loss.



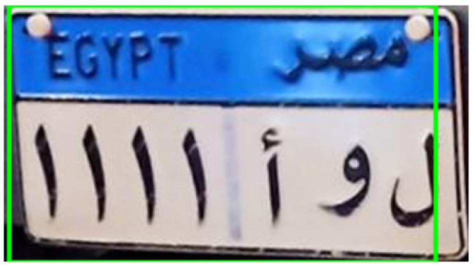
First model output (example 1)



Second model output (example 1)



First model output (example 2)



Second model output (example 2)

Figure 4.2: Detection comparison between the first model and the improved second model. The second model consistently captured the full plate boundaries that were missed by the first. See Figure 4.2.

4.2.5 Training Configuration

Final model hyperparameters:

- Epochs: 50
- Batch Size: 16
- Input Image Size: 640×640

- Optimizer: Adam
- Learning Rate: 0.001
- Weight Decay: 0.0005
- Early Stopping Patience: 10 epochs
- Cosine Learning Rate Scheduling: Enabled

The model training process was monitored using both the Precision-Recall (PR) curve and training loss metrics. As shown in Figure 4.3, the PR curve provides insight into the trade-off between precision and recall during evaluation. Additionally, Figure 4.4 illustrates the evolution of training loss and validation performance over the course of the training epochs.

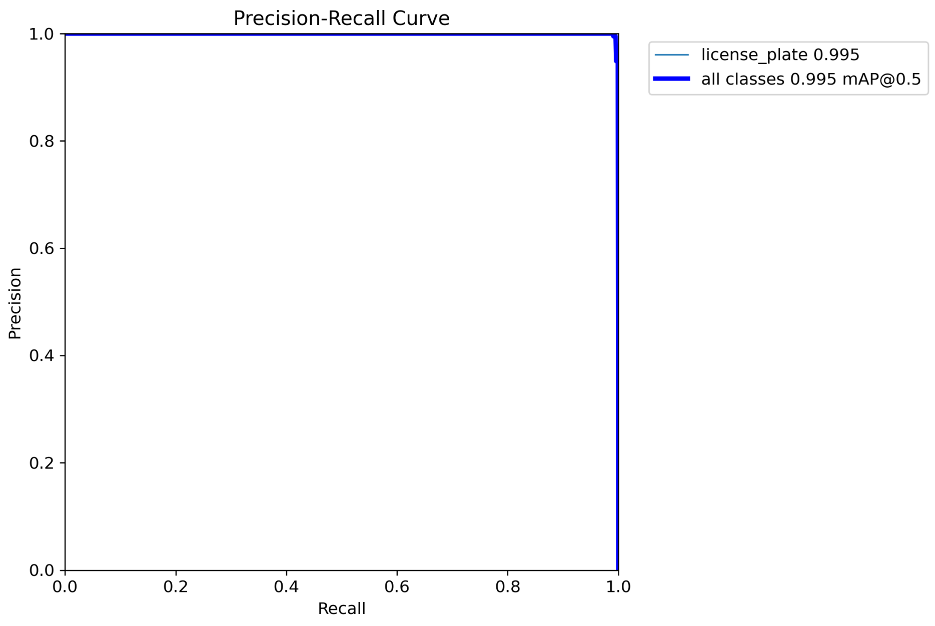


Figure 4.3: Precision-Recall (PR) curve for the final LPR model.

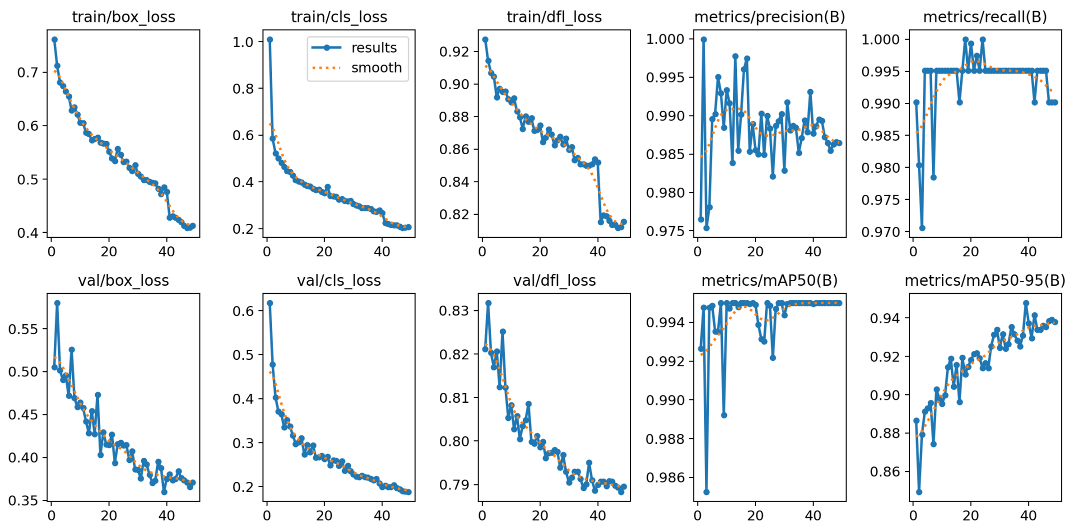


Figure 4.4: Training loss and evaluation metrics over epochs.

4.2.6 Evaluation and Insights

The final version of the LPR system demonstrated:

- High detection accuracy for Egyptian-style license plates under normal and moderately low lighting conditions, as well as in slightly blurred images.
- Improved localization performance achieved through data augmentation and optimized hyperparameter tuning.
- Accuracy was further enhanced by expanding the training dataset, which improved the model's ability to generalize across different plate styles and conditions.

However, the model still struggled to detect license plates that were **extremely blurred**, severely degraded in quality, or captured under very poor lighting, where key visual features were no longer distinguishable.

4.2.7 Challenges and Future Enhancements

Despite the model's strong overall performance, some limitations remain:

- Reduced detection accuracy when plates are viewed at extreme angles or are partially obstructed.
- Difficulty recognizing plates in cases of extreme blur or severely poor lighting, where critical visual features are no longer distinguishable.

To improve the system further, the following enhancements are proposed:

- Deploy the model on an embedded device (e.g., Raspberry Pi or Jetson Nano) to enable real-time gate control.
- Integrate object tracking to support robust detection in continuous video streams rather than individual frames.

4.3 Arabic Letters and Digits Recognition

4.3.1 Overview

To accurately interpret Egyptian license plates, which contain a mix of Arabic letters and numerical digits, a dedicated character recognition module was developed. Traditional Optical Character Recognition (OCR) tools, such as EasyOCR, did not provide reliable results due to limited generalization and inaccurate segmentation of Arabic characters. As a result, a custom object detection model was implemented using YOLOv8, where each Arabic letter and digit was treated as a separate object class.

4.3.2 Tools and Technologies

- **YOLOv8:** Object detection framework used to identify and classify Arabic letters and digits.
- **Roboflow:** For dataset annotation, class balancing, and preprocessing.
- **OpenCV:** For image manipulation and cropping.
- **Python, Jupyter Notebook:** Development and training environment.

4.3.3 Dataset Preparation

A custom dataset was created by manually cropping Arabic characters and digits from annotated license plate images. Care was taken to ensure that each character class (e.g., specific Arabic letters and digits) was well represented and labeled consistently. The final dataset was split as follows:

- **Training set:** 3,725 images
- **Validation set:** 745 images
- **Testing set:** 549 images

Each set was made mutually exclusive to avoid data leakage and ensure the model's ability to generalize to unseen samples.

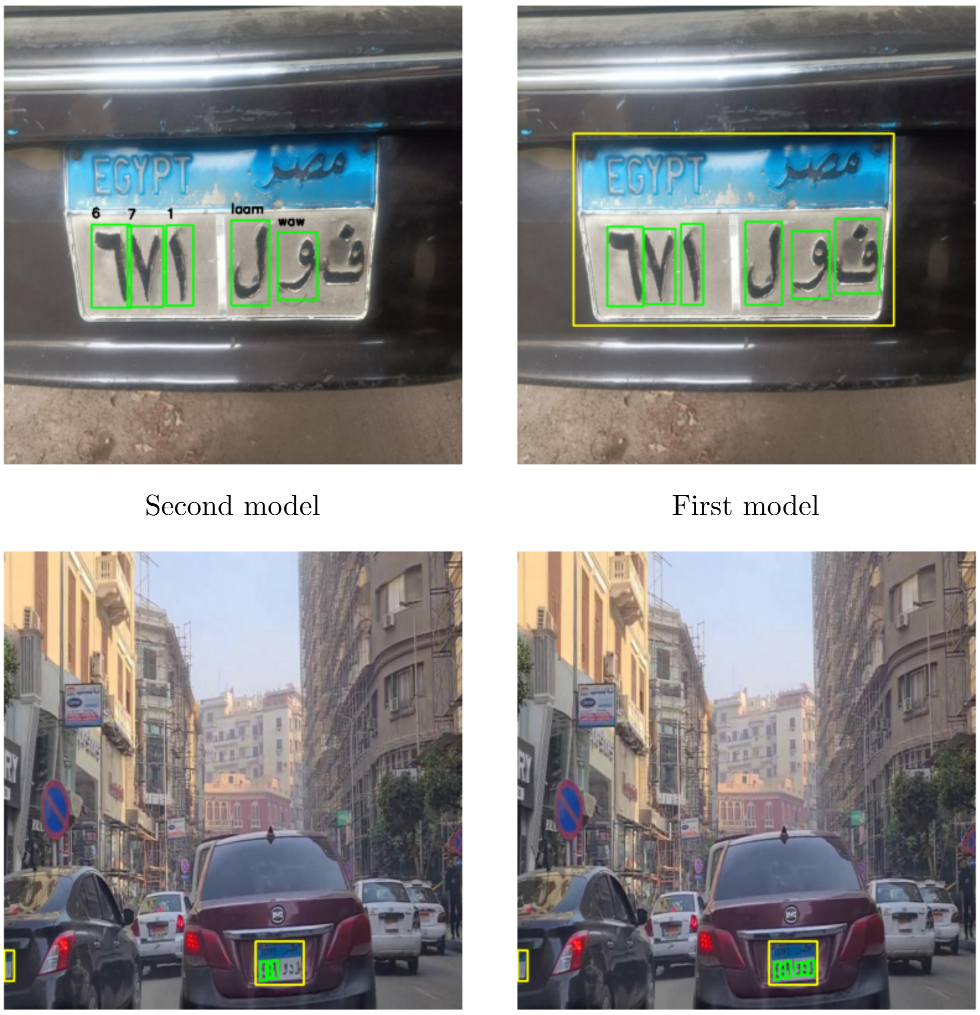
4.3.4 Training Process and Results

The development of the Arabic character recognition module was carried out in three stages:

Stage One: Fine-tuning EasyOCR. The first approach involved fine-tuning the EasyOCR framework using a publicly available dataset of Arabic letters and digits [9]. This dataset contains 2,481 samples for each Arabic character and digit, offering full coverage of the Arabic alphabet. While the model performed well on standard text images and PDF scans, it failed to generalize to Egyptian license plates. This limitation is likely due to differences in font style, background texture, and segmentation complexity between printed text and real-world plates. Standard OCR models are typically trained on clean, uniformly formatted text, and they often struggle with the noise, non-standard fonts, and spatial variability present in license plates.

Stage Two: Object Detection with YOLOv8. To overcome the limitations of OCR, the second stage implemented an object detection-based approach using YOLOv8. A labeled dataset containing 3,725 license plate images was used for training [10]. Each Arabic letter and digit was treated as an individual object class. This approach significantly improved detection accuracy and generalization, especially under real-world conditions such as variable lighting and different plate formats.

Stage Three: Dataset Expansion. In an attempt to further improve performance, the dataset was expanded with additional annotated images. However, this resulted in reduced model generalization and increased false detections. The expanded dataset introduced too many repetitive patterns with limited variation, causing the model to overfit and produce false positives. This outcome reinforced the importance of dataset quality and diversity over sheer volume. The results of both models are illustrated in Figure 4.5.



Second model First model

Second model First model

Figure 4.5: Performance comparison between second model (left) and first model (right). The second model produced false positives and missed characters, while the first model gave complete and accurate detections.

4.3.5 Training Configuration

Final model hyperparameters:

- Epochs: 100
- Batch Size: 16
- Input Image Size: 640×640
- Optimizer: Adam

- Learning Rate: 0.001

- Weight Decay: 0.0005

- Early Stopping Patience: 10 epochs

- Cosine Learning Rate Scheduling: Enabled

The training process for the Arabic character recognition model was tracked using both Precision-Recall (PR) curves and loss metrics. As shown in Figure 4.6, the PR curve captures the balance between precision and recall. Figure 4.7 displays the training loss and validation performance across epochs, helping to monitor model convergence and early stopping.

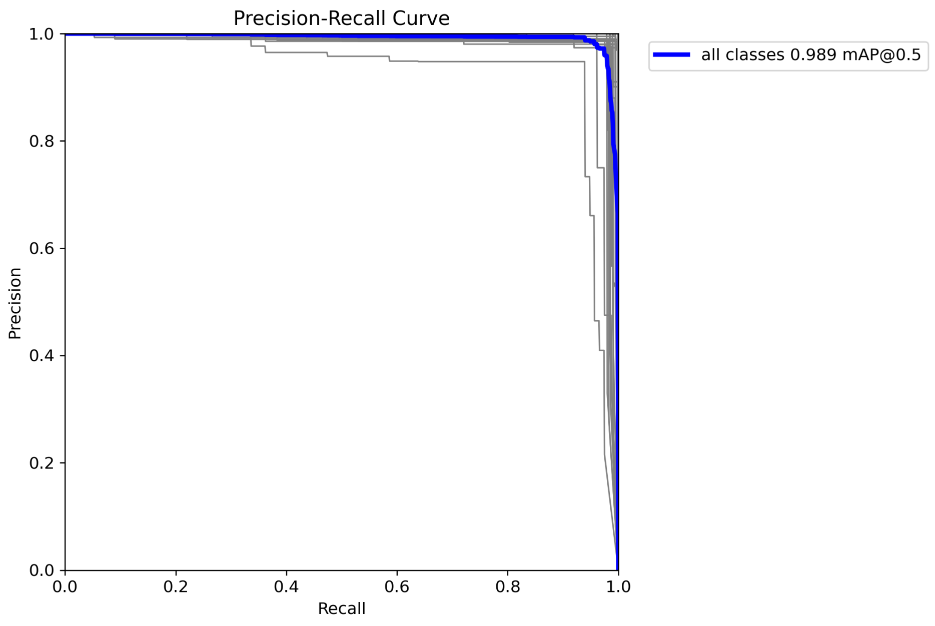


Figure 4.6: Precision-Recall (PR) curve for the final Arabic character recognition model.

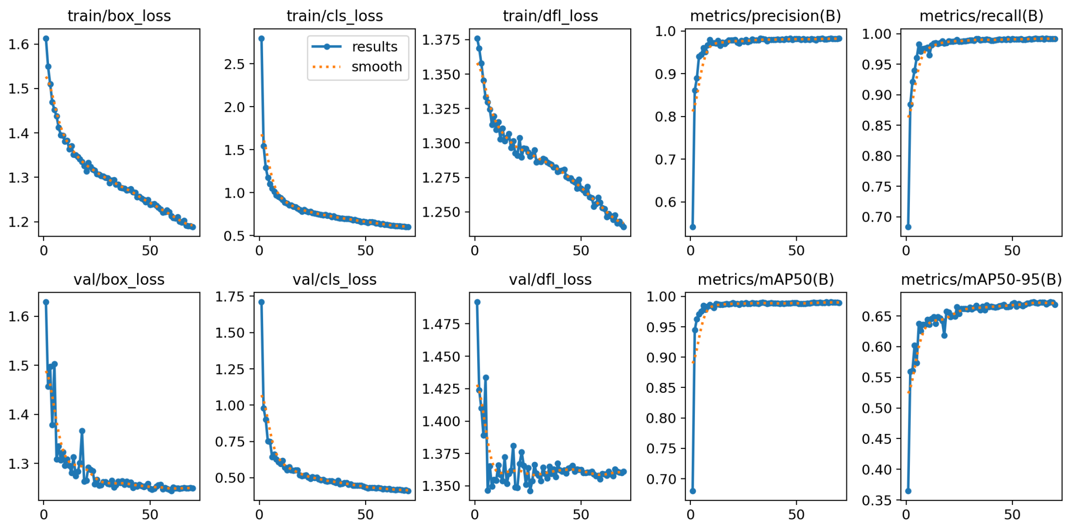


Figure 4.7: Training loss and evaluation metrics over epochs for the Arabic character recognition model.

4.3.6 Evaluation and Insights

The final version of the Arabic character recognition model demonstrated:

- Strong detection accuracy for Arabic letters and digits under a range of lighting conditions and plate styles.
- Improved robustness achieved by switching from traditional OCR to an object detection approach, which handled character segmentation more effectively.
- The initial model, trained on a smaller but cleaner dataset, generalized well and produced consistent results in real-world scenarios.

However, performance declined when the training dataset was expanded without adequate diversity. The second model became overly sensitive to recurring patterns, resulting in reduced generalization and increased false positives. As illustrated in Figure 4.5, the second model often missed or misclassified characters, whereas the first model delivered complete and accurate predictions.

4.3.7 Key Findings

- **EasyOCR Limitation:** Prebuilt OCR tools performed poorly with Arabic script, especially for detection and segmentation under real-world conditions.
- **Detection-Based Approach:** Treating each character as a separate object class using YOLOv8 proved more reliable than conventional OCR.

- **Overfitting Risk:** Dataset expansion led to performance degradation due to lack of intra-class variation and excessive repetition of similar character samples.

This analysis highlights a key insight: dataset diversity and quality matter more than sheer size when training character detection models.

4.3.8 Challenges and Future Enhancements

Several challenges were encountered during development:

- High similarity between certain Arabic letters increased the likelihood of misclassification.
- Real-world plate variations (e.g., font, spacing, background noise) added complexity.

To improve system robustness, future work should include:

- Enhancing the dataset with more diverse plate types and fonts.
- Adding image augmentation (e.g., blur, brightness shift, rotation) to improve model generalization.
- Deploying lightweight versions of the model on embedded systems for real-time inference.