Automatic Number Plate Detection and Recognition (ANPDR) system

Session: 2021 - 2025

Submitted by:

M. Yaqoob	2021-CS-118
Ahmed Raza	2021-CS-161

Supervised by:

Samyan Qayyum Wahla

Department of Computer Science

University of Engineering and Technology Lahore

Pakistan

Contents

1 ANPDR System and Goal 11

The Automatic Number Plate Detection and Recognition (ANPDR) system is related to the goal of sustainable cities and communities.

Goal 11: Sustainable Cities and Communities (from the United Nations' Sustainable Development Goals - SDGs):

- The goal of sustainable cities and communities includes making cities and human settlements inclusive, safe, resilient, and sustainable.
- ANPDR system contributes to the safety and security aspect of this goal. By automatically detecting and recognizing number plates, these systems can assist in various applications, such as monitoring and managing traffic, enhancing law enforcement, and improving overall urban security.

ANPDR is directly aligned with the goal of sustainable cities and communities. ANPDR system is primarily designed for traffic management, law enforcement, and urban planning, which fall under the broader umbrella of creating sustainable and well-functioning urban environments.

2 Overview

ANPDR system is a sophisticated application designed for the detection and recognition of license plates in images or video streams. It leverages the Ultralytics YOLO framework, a popular choice for real-time object detection, and integrates various Python modules for image processing, neural network operations, and GUI interactions

3 Libraries

3.1 YOLO (You Only Look Once)

3.1.1 Description

YOLO is a state-of-the-art, real-time object detection system. It's known for its speed and accuracy in detecting objects in images or video streams.

3.1.2 Functionality

Unlike traditional object detectors, which typically apply a classifier to various regions of an image, YOLO applies a single neural network to the whole image. This network divides the image into regions and predicts bounding boxes and probabilities for each region simultaneously.

3.1.3 Usage in ANPDR

In our system, YOLO is used for the crucial task of detecting license plates in images.

3.2 EasyOCR

3.2.1 Description

EasyOCR is a Python library that simplifies the process of Optical Character Recognition (OCR). It can read text from images and is capable of handling multiple languages.

3.2.2 Functionality

EasyOCR combines the power of deep learning models with the convenience of a simple API. It usually employs models like CRNN (Convolutional Recurrent Neural Network) for text detection and recognition.

3.2.3 Usage in ANPDR

After YOLO detects license plates, EasyOCR is used to extract and interpret the text (numbers and letters) from these plates.

3.3 Tkinter

3.3.1 Description

Tkinter is the standard GUI (Graphical User Interface) toolkit for Python. It provides an easy way to create simple GUI applications.

3.3.2 Functionality

Tkinter offers various widgets like buttons, labels, and text boxes to build a user interface. It's widely used for its simplicity and the fact that it's included with Python.

3.3.3 Usage in ANPDR

In our system, Tkinter is used to create the interface through which users upload images and view the recognition results.

3.4 PIL (Python Imaging Library)

3.4.1 Description

PIL, now known as Pillow in its maintained version, is a library in Python that adds support for opening, manipulating, and saving many different image file formats.

3.4.2 Functionality

It provides image processing capabilities such as resizing, cropping, color manipulation, and more.

3.4.3 Usage in ANPDR

PIL is used for image processing tasks, such as preparing images for detection and recognition processes.

3.5 PyTorch

3.5.1 Description

PyTorch is an open-source machine learning library used for applications such as computer vision and natural language processing. It's known for its flexibility and ease of use, especially in rapid prototyping.

3.5.2 Functionality

PyTorch provides two high-level features: tensor computation with strong GPU acceleration and deep neural networks built on a tape-based autograd system.

3.5.3 Usage in ANPDR

Likely used for neural network operations, including training and running inference with models like YOLO.

3.6 Hydra

3.6.1 Description

Hydra is a framework for elegantly configuring complex applications. It allows for a hierarchical configuration with the ability to dynamically create configurations from the command line.

3.6.2 Functionality

It simplifies the process of managing complex configurations, often required in machine learning projects.

3.6.3 Usage in ANPDR

Hydra might be used for managing the various configurations needed for the object detection and OCR components of our system.

3.7 Neural Networks (NN) in the Context of ANPDR System

Neural networks, particularly in the context of our Automatic Number Plate Detection and Recognition (ANPDR) system, are crucial for processing and interpreting complex visual data. Here's a brief overview of how neural networks are typically utilized in such systems:

3.7.1 Description

Neural Networks: At their core, neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling, or clustering of raw input.

3.7.2 Functionality

Pattern Recognition: Neural networks excel at pattern recognition, which is essential for tasks like image classification, object detection, and character recognition.

Learning from Data: They learn to perform tasks by considering examples, generally without being programmed with any task-specific rules. For instance, in an ANPDR system, a neural network can learn to identify and interpret license plates from a large dataset of various images.

3.7.3 Usage in ANPDR

Object Detection (YOLO): In our system, neural networks are the backbone of the YOLO model. They enable the model to detect license plates in images or video feeds by learning from a vast array of annotated training data.

Optical Character Recognition (EasyOCR): Another crucial use of neural networks in our system is through EasyOCR for character recognition. After detecting a license plate, the system uses a neural network trained to recognize and interpret the alphanumeric characters on the plate.

3.7.4 Importance in Computer Vision

Computer Vision Tasks: Neural networks have revolutionized the field of computer vision, providing the tools to perform complex tasks like object detection, image segmentation, and image generation.

Deep Learning: Modern neural networks, especially deep learning models, have layers of interconnected nodes that can capture hierarchical patterns in data. This feature is particularly beneficial in image-related tasks where different layers can detect various features, from simple edges in the initial layers to complex shapes and objects in deeper layers.

In summary, neural networks in our ANPDR system are fundamental to processing and understanding the visual data essential for detecting and recognizing license plates. Their ability to learn from data and identify patterns makes them indispensable in the field of computer vision and particularly in applications like our ANPDR system.

4 Application

4.1 main.py

The `main.py` script in our ANPDR system is a Python application with a graphical user interface (GUI), designed for license plate detection and recognition. Key components and functionalities include:

4.1.1 Overview

- **GUI Framework:** Uses `tkinter`, a standard Python interface to the Tk GUI toolkit.
- **Image Processing:** Interfaces with the Python Imaging Library (`PIL`) for handling images.
- **Threading:** Uses threading to run processes in parallel, enhancing performance and responsiveness.

4.1.2 Key Functionalities

- **GUI Components:** Creates a window with buttons and labels for user interaction.
- **Image Upload and Processing:** Allows users to upload images for license plate detection.
- **License Plate Recognition:** Imports and uses `predict_1.py` for the license plate recognition task.
- **Temporary File Management:** Handles creation and deletion of temporary files generated during recognition.

4.1.3 Detailed Functionality

- `run_func_with_loading_popup:` Runs a given process with a loading popup, using threading.

- `comment, extend, on_submit, remove_all_files, upload_file:` Functions handling various aspects of the GUI.

4.1.4 GUI Design

Defines a main window with buttons for uploading and submitting images, and labels to display messages and results.

4.1.5 Integration with External Script

The `get_text` function from `predict_1.py` is critical for processing uploaded images and extracting text from license plates.

4.2 predict_1.py

The `predict_1.py` script is a key component responsible for detecting and recognizing text from license plates. It integrates with Ultralytics YOLO and EasyOCR for OCR tasks.

4.2.1 Overview

- **Integration with Ultralytics YOLO:** Uses Ultralytics YOLO for real-time object detection.
- **EasyOCR for OCR Tasks:** Employs EasyOCR for Optical Character Recognition.
- **OpenCV (cv2) for Image Processing:** Utilizes OpenCV for image manipulations and processing.

4.2.2 Key Functionalities

- `ocr_image Function:` Extracts text from a detected region using EasyOCR.
- `DetectionPredictor Class:` Handles license plate detection using YOLO.
- `get_annotator, preprocess:` Methods for annotating and preprocessing images.
- `get_image_path, predict, get_text:` Functions for image path retrieval, prediction, and text extraction.

4.2.3 Integration and Workflow

- Designed to work with images in a specific directory.

- Applies YOLO for license plate detection and EasyOCR for text extraction.
- Writes extracted text to `output.txt`.

4.3 predictor.py

The `predictor.py` script contains the `BasePredictor` class, a central component of the Ultralytics YOLO-based detection system.

4.3.1 Overview

- **BasePredictor Class:** Foundation for creating and managing a YOLO object detection model.

4.3.2 Key Functionalities

- **Initialization (__init__):** Sets up the predictor with configuration options.
- **Preprocessing (preprocess):** Method for preprocessing images.
- **Annotator (get_annotator):** Creates an annotator for drawing bounding boxes.
- **Prediction Workflow (__call__):** Main method for running the detection process.
- **Image and Video Handling:** Methods for displaying and saving predictions.
- **Callbacks (run_callbacks):** System for running predefined or custom callbacks.

4.3.3 Integration with YOLO and Computer Vision

- Integrates with other components of Ultralytics YOLO.
- Uses OpenCV (cv2) for image and video processing.

4.3.4 Customization and Configuration

- Users can customize behavior through configuration options.

4.4 filter.py

The `filter.py` script is involved in post-processing the output of the detection model, including filtering out detections based on confidence scores and applying non-maximum suppression.

4.5 ultralytics Directory

The `ultralytics` directory is the core of the system, containing modules and configurations for object detection and neural network operations.

4.5.1 Hub Subdirectory

Contains modules for authentication, session management, and utility functions.

4.5.2 Models Subdirectory

Hosts various configurations of the YOLO model for efficient object detection.

4.5.3 NN Subdirectory

Manages neural network operations, including computational backend selection and module definitions.

4.5.4 YOLO Subdirectory

Contains YOLO model configurations and data loading modules, crucial for model training and inference.

5 Component Analysis

5.1 Utilities

Utility modules provide essential functions for image processing and file handling.

5.2 Session and Authentication

Modules handle user sessions and authentication, ensuring system security.

5.3 Neural Network Customization

Modules define custom tasks and neural network modules, tailoring the model to specific detection tasks.

5.4 Backend Optimization

`Autobackend.py` optimizes computational resource usage, enhancing system performance.

5.5 Configuration Management

`hydra_patch.py` and `default.yaml` streamline configuration management, making the system adaptable and easy to configure.

6 Methodology diagram

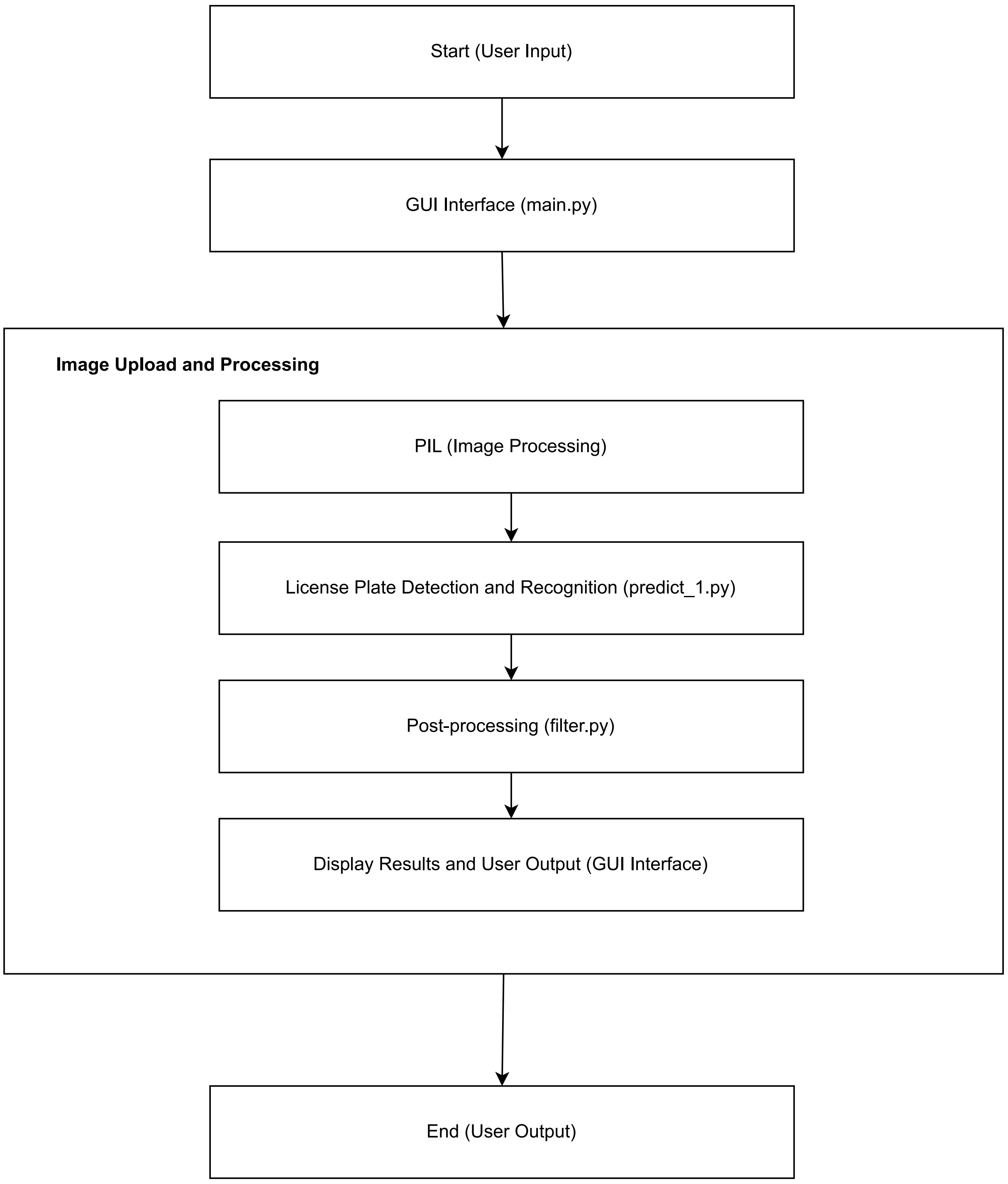
Start (User Input)
GUI Interface (main.py)
Image Upload and Processing
PIL (Image Processing)
License Plate Detection and Recognition (predict_1.py)
Post-processing (filter.py)
Display Results and User Output (GUI Interface)
End (User Output)

FIGURE 1: Methodology diagram

7 Application Areas

7.1 Traffic Management

ANPDR system is crucial for traffic monitoring and management. It will assist in enforcing traffic rules, optimizing flow, and reducing congestion through real-time data analysis.

7.2 Law Enforcement

In law enforcement, ANPDR system will aid in identifying and tracking vehicles involved in criminal activities. Automated license plate recognition will enhance the ability to locate stolen vehicles and those associated with suspicious incidents.

7.3 Parking Management

ANPDR can be used in parking facilities for monitoring vehicle entry and exit. Automated system can contribute to efficient space allocation and streamlined payment processing.

7.4 Security and Surveillance

The system is employed in security and surveillance applications to control access to restricted areas. ANPDR will enhance security by identifying and logging vehicles entering secure premises.

7.5 Toll Collection

ANPDR system is integrated into toll booths for automated toll collection. Vehicles passing through toll gates will be recognized, and toll fees are automatically deducted.

7.6 Smart Cities and Urban Planning

ANPDR will contribute to smart city initiatives by providing valuable traffic data. It will aid in urban planning, optimizing infrastructure, and improving overall city sustainability.

7.7 Logistics and Fleet Management

In logistics and fleet management, ANPDR can be utilized to track and manage vehicle movements. Fleet operators benefit from route optimization, vehicle monitoring, and schedule compliance.

7.8 Border Control and Customs

At border crossings and customs checkpoints, ANPDR can be employed to monitor and manage vehicle movement, assisting in the identification of vehicles requiring further inspection.

7.9 Public Safety and Amber Alerts

ANPDR system can be used in public safety initiatives, such as issuing Amber Alerts. It can contribute to identifying and tracking vehicles associated with emergencies or missing persons.

7.10 Environmental Monitoring

In environmental monitoring, ANPDR will collect data on vehicle emissions and traffic patterns, aiding in assessing the environmental impact of transportation systems.

7.11 Commercial Applications

ANPDR can find applications in various commercial settings, including automated car wash systems, where license plate recognition is linked to customer accounts.

8 Dataset Collection

The dataset used in this project was obtained by capturing pictures of license plates on roads. The process of collecting the dataset involved the following steps:

8.1 Location and Coverage

The dataset includes images captured from roads in Lahore, ensuring a diverse representation of license plates from different geographic areas. This geographical diversity contributes to the robustness and generalization of the ANPDR system.

8.2 Privacy Considerations

In the process of dataset collection, privacy considerations were taken into account. Steps were implemented to ensure that sensitive information, beyond license plates, was not inadvertently captured or included in the dataset. Additionally, efforts were made to comply with applicable privacy regulations and ethical considerations.

8.3 Dataset Size and Composition

The final dataset comprises 200 images of license plates, each accompanied by corresponding annotations. The dataset encompasses a variety of license plate designs, font styles, and vehicle types to enhance the diversity of training data.

9 Results

We trained model on 150 images and then tested it with 50 images. Results were 89% accurate.

10 Conclusion

The ANPDR system demonstrates a sophisticated approach to object detection and recognition, integrating various aspects of machine learning and computer vision. Its modular design and configurability make it a robust solution for automatic number plate detection and recognition.