# Number Plate Recognition

## AIM

To develop a robust and efficient number plate recognition that integrates real time detection,tracking and text extraction of license plates from video feeds. By using advanced deep learning techniques like YOLO for object detection and PaddleOCR for optical character recognition, it enhances the accuracy and performance in identifying number plates.

## HOW I STARTED

1. First I have researched number plate recognition from online sources.
2. Got a model file for a number plate, which contains 1 class(number plates).
3. Started with detection logic, used Ultralytics YOLO library and found Motpy(multi object tracker) model for tracking.
4. Test with the code and researched on OCR for text extraction on license plates.
5. Replaced Motpy model with YOLO tracking model, testing and writing code for text extraction using OCR.
6. Test EasyOCR for number plate text extraction, for better performance researched about LLM models.
7. Researched about PaddleOCR as it shows better performance.
8. Tested PaddleOCR with code and got better performance.
9. Preprocessed the images with paddle ocr and extracted text while preserving its original formatting.
10. Used python data structure queue to store the images after detection and tracking and to pass to ocr for extraction.
11. 3 files have been set up:
    - Detection and Tracking.py
    - Text Extraction.py

- Main.py(which integrates queue for multiprocessing)

## PACKAGES USED

- ☐ Cv2: Python interface for OpenCV, widely used for image processing and computer vision tasks.(reading, writing and manipulating images and videos, feature extraction and object tracking)
- ☐ YOLO: real-time object detection model known for its speed and accuracy.(tracking and classification of objects in images or video streams)
    - Single neural network: Unlike traditional methods that apply classifiers at different parts of the image, YOLO treats object detection as a single regression problem, predicting bounding boxes and class probabilities directly from full images.
    - Easy integration: The Ultralytics implementation allows for straightforward integration with Python projects, providing pre-trained models and tools for training custom models.
- ☐ OS: Operating system-dependent functionality like reading, writing to the file system.(allows for data manipulation tasks like creating directories or checking file existence)
- ☐ NumPy(np): handles multi-dimensional arrays, essential for image data manipulation. (often used to handle image data in array form)
- ☐ PaddleOCR:PaddleOCR is an open-source Optical Character Recognition (OCR) tool developed by PaddlePaddle, designed to recognize text in images. It supports multiple languages, including English, Chinese, French, German, Korean, and Japanese.
- ☐ Multiprocessing: This module in Python is a powerful library that allows to create and manage multiple processes, enabling parallel execution of code to improve performance
- ☐ Queue: It is a data structure that follows the FIFO (First-In-First-Out) principle, meaning that the first element added to the queue will be the first one to be removed.

- ☐ BoT-SORT (Basic Online Tracker with Simple Online and Realtime Tracking): An advanced tracking algorithm that combines the strengths of SORT with additional features for improved performance. It is designed to handle multiple object tracking in real-time scenarios.It works by maintaining unique identities for detected objects across multiple frames.

## WHY BOT-SORT?

BoT-SORT is more accurate than ByteTrack, especially in crowded situations. It has a MOTA(multi object tracking accuracy) score of about 80.5, while ByteTrack scores around 80.3. BoT-SORT handles occlusions better, whereas ByteTrack often loses track of objects in close interactions.

## WHY NOT EASY OCR AND PYTESSERACT?

Switching from EasyOCR to PaddleOCR is due to PaddleOCR's superior performance in accuracy, especially for complex text and multiple languages. PaddleOCR also offers better handling of non-90-degree rotations and occlusions.

- Speed: Pytesseract can be slower because it loads the Tesseract engine each time it processes an image, whereas PaddleOCR is optimized for faster performance.
- Accuracy: PaddleOCR generally provides higher accuracy and better results for various text types compared to Pytesseract, which may struggle with certain fonts or layouts.
- Flexibility: PaddleOCR supports a wider range of features and configurations, making it more adaptable for different OCR tasks.

## WHY CHANGED FROM MOTPY TO BOT-SORT(YOLO)?

1. Seamless Integration: YOLO's tracking is built into its framework, making it easier to use with object detection and simplifies the workflow.
2. Better performance in terms of accuracy and speed.

3. Ease of Use: It reduces the need for additional configuration and setup compared to integrating separate libraries like MotPy.

## WHAT ARE THE BENEFITS OF USING A QUEUE?

1. Faster Processing: Queues allow images to be processed in memory, avoiding slow file saving.
2. Real-time Handling: Enables immediate processing of images as they are generated.
3. Better Resource Management: Controls memory usage
4. Allows image capturing and saving to happen independently, improving responsiveness.
5. Flexibility: Supports batch processing and selective saving without needing to read from disk.

## DETECTION AND TRACKING.PY

1. Initialization:
    - Loads the YOLO model and sets up video capture and output directory.
    - Initializes counters for frame processing, detected plates, and their last known positions.

2. Frame Processing Loop:
    - Continuously reads frames from the video until the end is reached.
    - Applies YOLO tracking to detect objects in each frame.

3. License Plate Detection:

- For each detected object, check if it is a license plate.
- Extracts bounding box coordinates for the detected plate.
- Compares the current detection with previously detected plates to determine if it's a new plate or a previously seen one.

4. Tracking Logic:
    - If a new plate is detected, assigns a unique ID and initializes its detection count.
    - Updates the last known position of the plate and increments its detection count.

5. Bounding Box Drawing:
    - Draws a rectangle around detected license plates on the frame for visualization.

6. Cropped Image Processing:
    - If a plate is detected consistently for 15 frames, crops the image of the plate.
    - Calls a provided function to process the cropped image.
    - Saves both the full frame and cropped image (if enabled) to the output directory.

## **TEXT EXTRACTION.PY**

- Initializes the PaddleOCR model with English language support and angle classification.
- Creates a set to track processed texts, ensuring no duplicates are saved.

- Image Validation:
    - Checks if the image is loaded properly. If not, print an error message.
- Image Preprocessing:
    - Resizes the image to enhance OCR accuracy.
    - Converts the resized image to grayscale.
    - Applies Gaussian blur to reduce noise and improve text recognition.
- Text Extraction:
    - Uses PaddleOCR to perform OCR on the preprocessed image.
    - Iterates through the results and extracts recognized texts.
- Duplicate Check:
    - Checks if each extracted text has already been processed. If not, it adds the text to the set of processed texts and appends it to a list for saving.
- CSV Saving:
    - If any new texts are extracted, call the save_to_csv method to save them.
- File Existence Check:
    - Checks if the CSV file already exists to determine whether to write a header row.
- Writing to CSV:
    - Opens the CSV file in append mode and writes the extracted texts as a single row.
    - If it's the first write, it includes a header row with "Extracted Texts".
- Confirmation Message:
    - Prints a message confirming that data has been saved along with the extracted texts.

**MAIN.PY**

1. Initialization:
    - Creates a queue (cropped_images_queue) to hold cropped images for processing.

2. OCR Process:
    - Starts a separate process that runs the process_cropped_image method to handle text extraction from images.

3. Image Processing Loop:
    - Continuously checks the queue for cropped images.
    - If an image is available, it processes it using the OCR Processor
    - If it receives a None value, it stops processing.

4. Text Extraction:
    - For each cropped image, it uses the OCR processor to extract text.
    - Prints the extracted text along with its confidence score if results are found.

5. Plate Detection:
    - Initializes the PlateDetector to detect license plates in video frames.
    - Passes cropped images of detected plates to the queue for OCR processing.

6. Termination Signal:
    - After starting detection, it adds a None value to the queue to signal that processing is complete and to terminate the OCR process.
7. Process Management:
    - Waits for the OCR process to finish before exiting