IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

Smart Vehicle Identification and Tracking Using YOLO-Based License Plate Recognition

Ms. Aashita Shrishail Hirmukhe and Ms. Asha Mane
Sadhu Vaswani Institute of Management Studies for Girls ,Pune
Savitribai Phule Pune University, Pune
aashita656@gmail.com and ashamane90@gmail.com

Abstract: *Manual verification of vehicle registration plates can be time consuming and error-prone. This paper details a functional prototype which enables license plate detection, reading and results presentation. Once images or videos have been uploaded the system uses Python to detect license plates within them, and then extracts and displays the plate information through a simple web interface. Detection is done by training a YOLO model using the Ultralytics library. The license plate is cropped using OpenCV, and then the characters are read using EasyOCR. If a video is used, ByteTrack will link detections between frames using temporary tracking IDs. The system has been made using Gradio to create a dashboard for the user. A detection section allows users to upload an image and detect license plates. There is also a video tracking section, detection history section and CSV export. On one example test image the detection had an accuracy of around 90.1%. The OCR stage found the plate text "DL1CA6207 IND" with an accuracy of around 99.43%. These are only for one example and do not give an indication of the overall accuracy. This project contributes by providing a functional, open source solution to license plate detection, recognition, tracking and recording. The limitations are well discussed and include OCR errors, additional printing on the plates, temporary tracking IDs and there needs to be more testing on a larger scale. A clear plan is provided to properly evaluate the project.*

Keywords: automatic license plate recognition, YOLO, EasyOCR, OpenCV, ByteTrack, Object Tracking, Gradio, Vehicle Monitoring.

I. INTRODUCTION

Vehicle registration plates are the most prominent identifying feature on a vehicle. These are used when parking, within housing societies, toll plazas, campuses and controlled entries. If there are only a few vehicles then someone can manually identify and record the plate. However if there are a large amount of vehicles going through, this will become time-consuming, tedious and error-prone. The plates could be small, dirty, blurry or at an angle; also the plate will not be stored in a searchable database using a manual approach.

Automatic License Plate Recognition (ALPR) can help complete this by using computer vision. Object detection is performed first using an object detection model to locate the plate. Next an Optical Character Recognition (OCR) tool reads the characters on the plate. Finally if a video is being used, tracking is used to associate the detected plates. Whilst these tasks are often grouped together, they each have unique purposes. Detection determines where the plate is, OCR determines what is written on the plate and tracking will determine if the plates from different frames are the same object.

Open source libraries make it easy for a student to put this all together. Within this project, a trained YOLO model has been used alongside OpenCV, EasyOCR, ByteTrack and Gradio to create a browser-based system called Smart Vehicle Identification and Tracking Using License Plate Recognition. This system has been developed for learning purposes and will not be used as a security product. This paper will specify exactly what has been measured.

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-40347

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

II. PROBLEM STATEMENT

When trying to monitor a large number of vehicles manually plate checking would be difficult. A script for detecting the plate may be used however the majority will not have an intuitive interface, video tracking, history or export functionality. The problem to tackle is how to combine plate detection, text recognition, video tracking and basic recording into a useful system using free tools.

Another issue to consider is the accuracy of the system. Although a model can accurately detect the plate, during OCR the model can misread some characters or include extra text. In addition, the video tracker can lose the object and give it a new ID when it comes back into view. To overcome this the system should provide the user with possible readings so they can confirm the correct one, rather than rely on confidence scores or track IDs.

III. OBJECTIVES

- Identify license plates from the uploaded images using the trained weights from YOLO.
- Crop the license plate using OpenCV and pass it on to the text recognition process.
- Use EasyOCR to extract possible license plate text and show the confidence value.
- Process uploaded videos and track the detections between frames using ByteTrack.
- Create a Gradio interface including a dashboard, image detection section, video tracking and detection history section.
- Store the results in a Pandas Table during the session and provide the option to export as a CSV.
- Outline any limitations of the prototype and create an appropriate testing plan for future evaluation.

IV. RELATED WORK

4.1 Object Detection using YOLO

YOLO (You Only Look Once) is an object detection algorithm that performs object detection as a single regression problem. The bounding boxes and class probability can be predicted by a single neural network. This makes it fast enough to be used for video and real-time applications. Improvements to accuracy and speed have been made in more recent versions . The Ultralytics toolkit provides an easy way to train, predict and track. However, the weights used for this project were only to detect plates.

4.2 License Plate Recognition

ALPR systems will usually have multiple steps within. First detecting the vehicle/plate, then the characters on the plate and then processing the text. It has been shown that using a YOLO based detection system can create a robust real-time ALPR pipeline . Plate detection and recognition has been looked into for unconstrained situations . Both studies note that blurring, lighting, plate size, viewpoint and occlusion can have a large impact on the performance. This is why detection and OCR have been made into two clear steps.

4.3 Text Recognition

Traditional OCR engines such as Tesseract are used on scanned documents. More recent methods have been created using deep learning techniques such as the CRNN architecture. This architecture uses convolutional features and sequence modeling to identify text within natural scenes. The EasyOCR open source library uses these scene text recognition models to produce possible text and confidence scores. However, since it is a generic OCR engine it may also pick up extra markings on the image that are near to the registration number. It can also be confused by characters that look similar to each other such as 0 and O, and 8 and B.

4.4 Multi-Object Tracking

To keep track of the detections of the same object across different frames a simple Kalman filter and Hungarian algorithm can be used to track objects in real-time. DeepSORT adds appearance features which help reduce this issue. ByteTrack avoids dropping low confidence detections which helps with partially occluded objects. However, the IDs from any of these trackers are only temporary associations.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

4.5 Interface and Image Manipulation

OpenCV is used for image and video manipulation . Gradio creates an easy way to share your machine learning model through a browser.

This implementation is a simplified version of what is available in the literature. A YOLO detection system has been used along with EasyOCR for text recognition. A CNN has not been implemented for specific character recognition on the plate.

V. RESEARCH GAP AND MOTIVATION

There is not a new detection algorithm developed in this project. The main practical gap I have overcome is putting together existing free tools to create a single piece of software that can be used by anyone through their web browser. They can upload an image/video, see the annotated result, see the candidates, see a history table and export the data into a CSV without having to run multiple commands.

This project also helps to clarify some common misunderstandings. Firstly, there are two separate confidence levels: Detector Confidence and OCR Confidence. These should not be confused as a confidence level of how accurate the system is. It is possible to have a high confidence detection, but the plate may still be misread. Secondly, whilst having an ID within the tracker is helpful to keep track of one object within a video, there is no guarantee that it will be given the same ID if it goes out of frame or the video is run again. These points are important to consider when considering whether a system like this would be appropriate for use in real life.

VI. PROPOSED SYSTEM AND METHOD

A prototype system has been developed using Google Colab. Firstly, the detector itself was initialised from saved weight files. The image processing pipeline was implemented and tested without the GUI elements. Finally, the GUI was developed and video tracking, temporary history table and CSV export were implemented. Fig 1 shows an overview of the workflow. Below is a breakdown of each part of the development process.

Fig. 1. Workflow of the proposed prototype.

1. Take Input: Upload a JPG/PNG image or supported video file via the Gradio interface.
2. Detect: The YOLO model predicts the bounding box of potential license plates and a confidence score.
3. Crop: The predicted bounding box co-ordinates are used by OpenCV to crop the license plate out of the original frame.
4. Read Text: The cropped image is passed into EasyOCR which will return one or more potential text candidates and an OCR confidence score.
5. Track Video: If a video has been uploaded the ByteTrack algorithm will be used to track the detections between frames and assign them temporary track IDs.
6. Display: The annotated image/video will be displayed along with a table showing potential readings.
7. Log Data: Each detection is added to an in memory Pandas DataFrame which can be exported as a CSV file.

The confidence scores from both detections and OCR will be stored and displayed so the user can see where there is uncertainty.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

VII. TOOLS & IMPLEMENTATION

7.1 Python & Google Colab

Python was used throughout this project. A google colab notebook was used with optional GPU support within the cloud. The model weights were saved into Google drive.

7.2 YOLO Detector & OpenCV

The Ultralytics YOLO model provides the bounding box coordinates and confidence score for each plate region. Using the coordinates it uses OpenCV to crop the relevant area. The detector uses weights that have been trained to detect license plates. This model was trained using [insert data set name] data with [insert number] training images and [insert number] epochs.

7.3 EasyOCR

A reader was created once for the EasyOCR model. This means it isn't reloaded each time. The output from this model is just a candidate solution and not actual registration numbers. From the example shown below, the output had 'IND.' at the end. This is due to this being part of the marking on the plate. Many high security plates in India have an 'IND' marking. This shows how general OCR tools can include extra text.

In future an extra step could be added to clean the text. For example the general format of an Indian plate is 2 letters for the state, 1 or 2 digits for the district, 1-3 letters for the series and 4 digits. An example would be DL 1 C A 6207. However this is not included in the current version of the implementation. These rules would need to be carefully tested to ensure they don't delete any valid characters.

7.4 ByteTrack & Gradio

Object tracking between frames. Simple Kalman filter plus Hungarian algorithm can be used for real time object tracking. DeepSORT provides additional appearance features to reduce identity switches. ByteTrack also does not filter out low confidence detections which helps with partially occluded objects. However, the ID's provided by any of these trackers are temporary associations.

7.5 Implemented features

Feature	Status in the prototype
Plate detection in images	Implemented (YOLO)
Plate cropping	Implemented (OpenCV)
Candidate text with confidence	Implemented (EasyOCR)
Video tracking with temporary IDs	Implemented (ByteTrack)
Dashboard and detection history	Implemented (Gradio, in-memory table)
CSV export	Implemented (Pandas)
Permanent database	Not implemented
Live camera / CCTV input	Not implemented
Plate-format validation	Not implemented

Table 1. Features of the prototype

VIII. EXPERIMENTAL RESULTS

8.1 Demonstration result

For the demonstration phase, an image was used to show the system can successfully identify the licence plate area (YOLO confidence ~ 90.1%). EasyOCR can be used to identify the license plate text, in this case it identified

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

"DL1CA6207 IND" with a confidence of ~ 99.43%. These values relate only to this particular example, and do not indicate any estimates on the overall accuracy of detection/plate recognition.

Fig. 2. Preliminary single-image demonstration showing YOLO plate detection and EasyOCR candidate text. Confidence values are specific to this example and should not be read as overall system accuracy.

Evaluation item	Observed result	Interpretation
Images tested	1 demonstrated image	Preliminary example only
YOLO detection confidence	≈ 90.1%	Confidence for the detected region in this example
EasyOCR candidate text	DL1CA6207 IND	Candidate; must be compared with the real plate
EasyOCR confidence	≈ 99.43%	Confidence for this run, not accuracy
Full-plate exact-match accuracy	Not measured	Needs a labelled test set
Character accuracy	Not measured	Needs ground-truth plate text
Video tracking / ID switches	Not measured	Needs tests on recorded videos

Table 2. Demonstration result and evaluation status.

8.2 Discussion

The example demonstrates that the core image processing pipeline is able to find the plate location, and extract a candidate text. However, it does highlight two issues. Firstly, the OCR process has identified "IND" along with the registration number, so the text isn't clean enough to be stored yet. Secondly, despite having a high confidence in the OCR, there is text on the plate that is not part of the registration number. This confirms that confidence in the OCR is purely how confident the model is with what it has read. If the result is important it should always be checked by a human. It also highlights that the OCR isn't perfect, and can return text that is unrelated to the plate number.

This single example does not demonstrate how the system performs with different vehicles, lighting conditions, viewpoints, motion blur, plate designs, and partial obscuring. Therefore this result is just a qualitative example.

8.3 Evaluation Plan

A proper evaluation would involve using a test set that contains plate images with known text, these should not be any images used for training. The following metrics will be recorded:

Detection Accuracy: Precision, Recall and F1-Score by comparing predicted and labelled boxes (For example using an intersection over union threshold of 0.5).

Accuracy of full plate: Percentage of images where the text from the OCR is exactly the same as the true plate number.

Accuracy of characters: A percentage of the characters that are correct, can be calculated using edit distance.

Tracking Video: Number of vehicles, missed detections, changes in tracking ID, and frame rate.

Results for different conditions: Results for different times of day, night, blur, and different camera angles. This will show how the system performs poorly.

These metrics have not yet been performed on the full pipeline and will be reported once testing has taken place.

IX. LIMITATIONS

OCR can sometimes be confused by similar looking characters, this can cause some additional text to be picked up from the printed plate. For example with the IND image.

Quality of detection will depend on how blurry the image is, low lighting level, size of plate, angle of camera and how much of the plate is covered.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, October 2026 Impact Factor: 8.2

The ByteTrack IDs are only temporary, if the vehicle is lost and then detected again the ID will change.
The detection history is also stored in memory so will be lost if the notebook or application is restarted.
The results given are only based on one image, there isn't enough data to be able to say that it has good overall accuracy.
There is no form of text filtering or plate formatting validation applied to the OCR results in this version.
The prototype isn't linked to a long-term database, live camera and official vehicle registry.

X. PRIVACY AND RESPONSIBLE USE

Vehicle registration numbers could be used to identify individuals. This means that any data relating to plates should be considered personal data. Data should only be captured if there is appropriate consent and for a clear purpose. Test data should be kept secure and only a few people should have access to the CSVs exported. Unnecessary duplicates should be removed. As the system may contain errors it shouldn't be used as the sole factor in making enforcement or access control decisions or decisions with significant impact on individuals. Any actual usage would need to comply with data protection legislation in the country it would be used in.

XI. FUTURE SCOPE

Larger and more diverse set of plates can be implemented to test the system. Data can be added to give different lighting, angles and plate sizes. This data can be discussed in section 8.3.
Text cleansing and plate formatting can be added to the testing, this will help remove excess marks and reduce character confusion.
A longer video can be used to test the tracker. The number of missing tracks and ID switches can be recorded.
A database can be implemented such as SQLite to store the data.
Live camera functionality could be implemented.
A specific character recognition model for plates could be used, for example a CNN trained on characters from plates. This could be used instead of a general OCR.
Access control can be added to the system and the exported data stored securely.
All new features should be tested and documented.

XII. CONCLUSION

This paper outlines a prototype license plate detection and candidate text recognition application, developed using a mixture of YOLO, OpenCV, EasyOCR, ByteTrack and Gradio. The application includes all of the steps involved, allowing the user to upload media, view annotated outputs and save session data. For one image shown, there was a high detection confidence of around 90.1% and OCR confidence of around 99.43%. This is specific to that image. This work demonstrates how different open source computer vision tools can be used together to create a functional application. However, there are several issues that need to be addressed before any real deployment can occur. There were issues with OCR accuracy, additional text in print and temporary tracking IDs. A thorough evaluation should also be done.

REFERENCES

[1] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection, in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR).
[2] Redmon, J., and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv:1804.02767.
[3] Silva, S.M. and Jung, C.R. (2018). License Plate Detection and Recognition in Unconstrained Scenarios, in Proc. European Conference on Computer Vision (ECCV).
[4] Smith, R. (2007). An Overview of the Tesseract OCR Engine, in Proc. International Conference on Document Analysis and Recognition (ICDAR).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

[5] Shi, B., Bai, X. and Yao, C. (2017). An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and its Application to Scene Text Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence, 39(11).
[6] JaidedAI (2020). EasyOCR. GitHub Repository. Available at: https://github.com/JaidedAI/EasyOCR
[7] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple Online and Realtime Tracking, in Proc. IEEE International Conference on Image Processing (ICIP).
[8] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple Online and Realtime Tracking with a Deep Association Metric, in Proc. IEEE International Conference on Image Processing (ICIP).
[9] Zhang, Y., et al. (2022). ByteTrack: Multi-Object Tracking by Associating Every Detection Box, in Proc. European Conference on Computer Vision (ECCV). Available at: https://arxiv.org/abs/2110.06864
[10] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools. Available at: https://docs.opencv.org/
[11] Abid, A., et al. (2019). Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. arXiv:1906.02569.