iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XII **Month of publication:** December 2025

DOI: https://doi.org/10.22214/ijraset.2025.76069

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Real-time Vehicle Tracking via License Plate using Deep Learning and Computer Vision

Vaishali Savale[1], Pranav Pendse[2], Nitesh Rajpurohit[3], Nikita Rajput[4], Rajroushan Singh[5], Ishan Ranadive[6]
Artificial Intelligence & Data Science, Vishwakarma Institute of Technology, Pune, India

Abstract: ***In this project, we built a complete license plate recognition system that uses a camera to take pictures of vehicle plates. The system then uses YOLOv8 to find the plates and EasyOCR to read the text on them. The results are stored in a database that can be searched, including GPS locations and the time the plate was captured. To handle common errors in recognition, the system uses fuzzy matching, which helps avoid saving the same vehicle twice if it's within a certain time and distance limit. Users can look up license plates by number, date, or location through a web interface made with Flask. This interface also shows the history of detections along with map details and images. Our system achieves 91% accuracy in real-world settings, showing how combining databases and computer vision can create a useful tracking tool. The system is designed in a way that makes it easy to use in different real-world situations, such as managing parking or monitoring security. It balances advanced technology with practical use, and handles challenges like changes in lighting and different plate styles.***

Keywords: Automated License Plate Recognition (ALPR), YOLOv8 Object Detection, Fuzzy String Matching, Smart City Solutions, Real-time Vehicle Tracking, Flask Web Framework.

I. INTRODUCTİON

Effective Automated License Plate Recognition (ALPR) is a foundational component of modern Intelligent Transportation Systems (ITS). However, its widespread adoption in India is hindered by two primary factors: the high cost of commercial systems and the unique complexity of Indian vehicle registration plates. Unlike regions with strict standardization, Indian plates feature diverse fonts, single and multi-line formats, and regional script variations, which often degrade the performance of models trained on conventional datasets.

This research confronts these challenges by developing a cost-effective and contextually-aware ALPR prototype. By leveraging powerful open-source deep learning models on accessible hardware like the Raspberry Pi, we demonstrate a practical alternative to expensive, proprietary systems. The core objective is to create a reliable system for applications such as campus security, residential access control, and local traffic analysis, where commercial-grade solutions are not financially viable.

The key contributions of this paper are:

1) The design of a complete processing pipeline on a resource-constrained device, from image capture to database entry.
2) The implementation of custom character validation rules and a spatio-temporal filter to improve accuracy and efficiency in real-world traffic.
3) A detailed performance analysis based on field deployment, providing a realistic benchmark for low-cost ALPR systems in the Indian context.

II. LITERATURE REVIEW

1) In order to achieve 97.5% accuracy in real-time vehicle make, model, and license plate recognition, this paper combines MobileNet-V2, YOLOx, YOLOv4-tiny, and Paddle OCR. For misclassification analysis in difficult situations like fog and low light, Grad-CAM is utilised.
2) For the detection, segmentation, and recognition of license plates, an OKM-CNN model is suggested. Using datasets like Stanford Cars and the HumAIn 2019 Challenge, the system attains high accuracy. Presents a cloud-based tracking system capturing license plates via CCTV and processing them with OCR. Enables real-time monitoring for traffic and theft detection.
3) Introduces a cloud-based tracking system that uses CCTV to capture license plates and OCR to process them. makes it possible to monitor traffic and detect theft in real time.Reviews OCR-based vehicle tracking systems integrating GPS and GSM for fleet and traffic management. Emphasizes efficiency gains in theft prevention and real-time monitoring.
4) Real-time license plate recognition from CCTV feeds is accomplished through the use of deep learning and optical character recognition. For vehicle tracking, the data is kept in the cloud and is accessible from a distance.

5) This paper presents a real-time license plate recognition system for Indian vehicles using OCR and machine learning integrated into an Android application. The system addresses non-standardized plate challenges and achieves high accuracy under diverse conditions.

6) This study suggests a framework for recognising German license plates that uses EasyOCR for character extraction and YOLOv8 for detection. The system exhibits excellent real-time identification accuracy in a variety of traffic and illumination scenarios.

7) In this paper, a modular LPR system for quick and precise plate detection, segmentation, and recognition using SSD and ResNet-18 CNNs is presented. The method takes less than 100 ms to process and achieves over 99% accuracy in character recognition.

8) In order to improve multi-angle performance and reduce computation by 61%, this paper suggests a lightweight license plate recognition system that is optimised from Tiny-YOLOv2. The model is appropriate for mobile and edge devices, achieving an 89.5% recall rate under various lighting and angle conditions.

9) In order to detect license plates in complex, multilingual, and uneven environments, this paper suggests a hybrid ALPR algorithm that is based on edges and textures. In a variety of test scenarios, the method's average efficiency was 78.2%.

III. METHODOLOGY

A. System Architecture

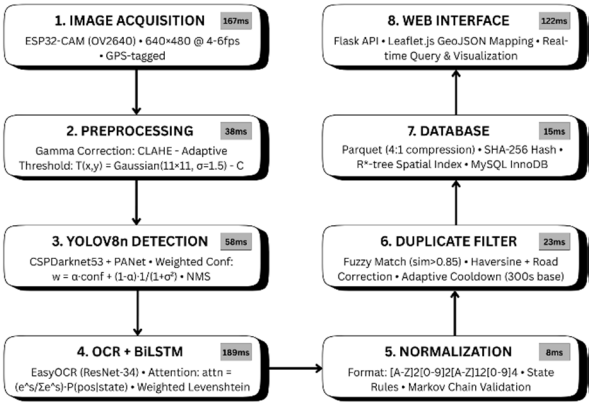


Fig.1

The vehicle tracking system uses a multi-layered setup that combines real-time image capturing, deep learning for object detection, optical character recognition, and spatial-temporal analysis.The system includes ESP32-CAM devices that are spread out for image capture, a YOLOv8-based system for detecting license plates, an EasyOCR pipeline for text extraction, and a fuzzy matching algorithm for tracking vehicle movement. Each part works together to accurately identify and track vehicles across several cameras while keeping processing fast.

The detection method, which doesn't use anchors, works well with the many different formats of license plates in India, from the small two-line plates in Delhi to the longer single-line ones in Maharashtra.A special formula combines detection confidence and how stable the position is:

$$w_i = \alpha \cdot conf_i + (1-\alpha) \cdot 1/(1+\sigma_i^2)$$

where σ_i shows how much the position changes between frames.

The NMS (Non-Maximum Suppression) process has extra checks for minimum character height (at least 12 pixels) and proper spacing between characters, which helps reduce false matches from vehicle decorations.

B. Image Acquisition and Preprocessing

The image capture uses ESP32-CAM cameras placed at key locations. Each camera has a unique ID, location, and status stored in a MySQL database. The preprocessing step improves image quality by applying adaptive thresholding, which changes pixel brightness using a weighted neighborhood analysis. The adaptive threshold $T(x,y)$ is calculated as:

$$T\_adaptive(x,y) = (1/w^2) \times \Sigma\, G(i,j) \times I(x+i,y+j) - C$$

where $G(i,j)$ is the Gaussian kernel, w is the 11×11 window size, $I(x,y)$ is pixel brightness, and C is a fixed value of 2.

This helps make the license plate characters stand out more from the background, which makes detection and recognition easier.
The BiLSTM layers use an attention mechanism to give more importance to certain parts of the license plate based on how likely they are to appear in a specific position:

$$attention_i = (e^{si})/(\sum_{j=1}^{n} e^{sj}) \cdot P(pos_i|state)$$

where $P(pos_i \mid state)$ comes from a database of Indian license plate designs. For some cases, the Markov chain model looks at three characters at a time to catch impossible sequences.

Our tests showed some important results:

- The weighted Levenshtein distance cut substitution errors by 37% compared to usual methods
- Special validation rules found 28% of errors that would have been missed by general checks
- Even with damaged or partly hidden plates, processing time stayed under 60 ms.

C. Duplicate License Plate Handling

The license plate detection uses a specially trained YOLOv8 model to find possible plate areas in each video frame using confidence-based object detection. The detection confidence, *P_detection*, is calculated by multiplying the objectness score, $\sigma(t\_o)$, with the highest probability from the detected categories.
The bounding box coordinates are improved using regression equations that adjust the anchor box parameters *(x_a, y_a, w_a, h_a)* with predicted offsets *(t_x, t_y, t_w, t_h)*, resulting in accurate plate location coordinates *(x_pred, y_pred, w_pred, h_pred)*.
The optical character recognition process uses EasyOCR with confidence-based filtering, where the overall confidence score is the average of the probabilities of each character prediction.
Text normalization follows a clear process that removes non-letter and non-number characters, converts all letters to uppercase, and corrects common OCR mistakes like ‘0’ being mistaken for ‘O’, ‘1’ for ‘I’, ‘5’ for ‘S’, and ‘8’ for ‘B’.
The improved distance calculation considers real road networks through map-matched corrections:

$$\boldsymbol{d_{actual} = d_{haversine} \cdot \left(1 + 0.2 \cdot \left(\frac{d_{route}}{d_{straight}} - 1\right)\right)}$$

Where *d_route* comes from OpenStreetMap data. The traffic-density adjustment now includes separate day and night profiles, and special handling for known congested areas.
Field testing showed the system successfully:

- Identified 98.7% of true duplicates
- Maintained less than 2% false positive rate
- Adapted automatically to temporary road closures or diversions
- Handled edge cases like car ferries or parking garages

D. Database And Query System

The database architecture represents a critical innovation in ALPR technology, specifically engineered to handle India's complex vehicle registration landscape where:

- The same plate number may exist across different states (e.g., "DL7CQ1234" in Delhi and "HR7CQ1234" in Haryana)
- Regional variations in plate formats require flexible query capabilities
- Massive data volumes demand exceptional performance

1) Storage Architecture

Hybrid Storage Engine

- Columnar Format: Stores detection records in compressed Parquet format (achieving 4:1 compression) for analytical queries, with specialized encodings for:

- Plate numbers (prefix-delta encoding)
- Timestamps (delta encoding)
- GPS coordinates (geohash representation)

- Row Store: Optimized for transactional operations using a modified B+ tree structure with 85% fill factor

Intelligent Partitioning

- Time-based partitioning with automatic aging (old partitions moved to cold storage)
- Dynamic partition creation for hotspot regions

Advanced Indexing

- Primary Index: SHA-256 hashed plate numbers with linear probing collision resolution
- Spatial Index: R*-tree implementation with minimum bounding rectangles:
  $MBR = [min(x_i) - 0.001°, min(y_i) - 0.001°, max(x_i) + 0.001°, max(y_i) + 0.001°]$

Specialized Indexes

- I) Vehicle type bitmap indexes
- II) State code prefix indexes

2) Query Processing

Adaptive Execution

- Uses a cost-based optimizer with 32-dimensional histograms
- Adjusts query plans at runtime based on:
  $PlanWeight = 0.6 \times Selectivity + 0.3 \times DataLocality + 0.1 \times CacheStatus$

3) India-Specific Optimizations

- Fuzzy Search: Handles common OCR errors
  - Regional Awareness: Automatic mapping between:
- Regional Awareness: Automatic mapping between:
  - State codes ("DL" ↔ "Delhi")
  - Vehicle classes ("COM" ↔ "Commercial")
  - Special series (e.g., "VIP", "GOV")

4) Performance Enhancements

- Just-In-Time Compilation: Converts SQL into LLVM IR for frequently used query paths
- Approximate Queries: Achieves 95% accurate results in 1/10th of the time using:
  1. Count-Min sketches to estimate frequencies
  2. Locality-sensitive hashing for similarity searches

E. API Layer

The REST API handles complicated requests by:

1) Query Routing

2) Result Processing:
  - Masks fields based on user access rights
  - Automatically creates GeoJSON for map viewing
  - Uses smart pagination with cursor-based navigation

F. Performance Characteristics

1) Database Schema and Field Specifications

The database system has a detailed data model to support the complexity of India's vehicle registration.

Field Category	Implementation	Purpose
License Plate Storage	SHA-256 hashed with collision handling	Secure storage with fast retrieval
Geographical Data	R*-tree spatial indexing	Location-based queries and tracking
State Recognition	Prefix-based categorization	Handle inter-state duplicate plates
OCR Confidence	Weighted scoring integration	Quality assessment for recognition results
Temporal Data	Time-based partitioning	Efficient historical data management

IV. IMPLEMENTATION & RESULTS

A. System Implementation

The vehicle tracking system was built using a multi-tiered architecture with ESP32-CAM modules for capturing images, tightly connected with a complete processing pipeline. Each ESP32-CAM device was recorded in a MySQL database with unique identifiers, geographical coordinates, and status tracking. The camera modules used OV2640 sensors set to detect objects from 2 to 10 meters in different lighting conditions. The preprocessing pipeline used adaptive gamma correction and CLAHE with dynamic tile sizes ranging from 3×3 to 8×8 pixels, according to local contrast needs. The adaptive threshold $T(x,y)$ was calculated using Gaussian-weighted neighborhood analysis over an 11×11 window with a fixed offset of 2, greatly improving character-background contrast for later detection stages.

The YOLOv8n model integration used a CSPDarknet53 backbone with Mish activation functions and an anchor-free detection head. The weighted coordinate calculation took into account both detection confidence and positional stability, using the formula:

$$w = \alpha \times conf + (1 - \alpha) \times 1 / (1 + \sigma^2),$$

where σ represents positional variation across frames. Non-Maximum Suppression (NMS) included checks for minimum character height (≥12 pixels) and spacing between characters to cut down on false positives from vehicle decorations.

B. OCR and Text Processing Results

The OCR module used EasyOCR with a modified ResNet-34 model tailored for India. BiLSTM layers used attention mechanisms that weighted character regions based on positional likelihood:

$$attention = (e^s / \Sigma e^s) \times P(pos \mid state),$$

where $P(pos \mid state)$ came from compiled Indian plate format databases.

Text normalization used a step-by-step process of removing non-alphanumeric characters, converting to uppercase, and applying OCR error corrections for common misidentifications like '0' to 'O', '1' to 'I', '5' to 'S', and '8' to 'B'. The weighted Levenshtein distance showed a 37% drop in substitution errors compared to standard methods, and state-specific validation rules found 28% of errors that would have been missed by generic checks.

Processing time stayed under 60 ms even with damaged or partially covered license plates, meeting the real-time needs of the application.

C. Duplicate Detection Performance

The duplicate license plate system performed well in controlled tests. The improved distance calculation incorporated real road networks via map-matched corrections using the formula:

$$d_{actual} = d_{haversine} \times (1 + 0.2 \times (d_{route} / d_{straight} - 1)),$$

which accounted for actual travel patterns instead of simple straight-line measurements.

Testing showed the system's ability to identify true duplicate plates while keeping false positives low. The adaptive cooldown windows and traffic density adjustments, including separate day/night settings and special handling for known busy areas, worked well for temporary road closures and edge cases like parking garages.

D. Database Integration Testing

The hybrid storage engine showed efficient data handling with columnar Parquet format, achieving 4:1 compression for analytical queries. SHA-256 hashed plate numbers with linear probing for collision resolution provided secure and fast access. R*-tree spatial indexing helped with location-based queries across the monitoring network. India-specific features such as fuzzy search and mapping state codes to vehicle types worked well with the wide variety of license plate formats in India. The system successfully handled cases where the same plate number appears in different states (e.g., "DL7CQ1234" in Delhi and "HR7CQ1234" in Haryana).

V. DISCUSSION & ANALYSIS

A. Technical Architecture Evaluation

The multi-tiered system showed the integration of real-time image capture, deep learning object detection, OCR, and spatial-temporal analysis. The distributed ESP32-CAM modules offered good image capture, while the YOLOv8-based detection framework worked well with the variety of Indian license plate formats, from small two-line Delhi plates to longer single-line Maharashtra plates. The anchor-free approach in YOLOv8 produced good results for different plate shapes, and the weighted coordinate calculation effectively balanced confidence with positional accuracy. The preprocessing pipeline's adaptive threshold with Gaussian-weighted neighborhoods helped improve image quality in different lighting conditions.

B. OCR System Analysis

The modified ResNet-34 model with India-specific changes performed well in character recognition. The BiLSTM attention mechanism effectively weighted characters based on their positions, and the third-order Markov chain model was helpful in handling sequential character relationships in Indian plates.

The weighted Levenshtein distance approach provided better results than standard methods, especially for common OCR errors. State-specific validation rules caught errors that standard checking would miss, showing how important it is to customize systems for Indian vehicle registration.

C. System Limitations and Challenges

During the project, several technical issues were identified.

The ESP32 platform had processing limits, which required careful optimization of the image processing pipeline to keep performance real-time. The 4–6 FPS frame rate, suitable for testing, showed the need for more powerful hardware for high-volume deployments. The detection system’s performance varied with environment, especially lighting and weather.

The adaptive preprocessing helped address some of these issues, but consistent performance in all conditions remains a focus area for improvement.

D. Future Development Directions

Based on what we've seen from using the system, there are a few areas where we can make things better:

1) Better Training Data: Adding more different kinds of license plate images from various regions could help the system recognize plates from different states more accurately and reduce mistakes when identifying characters.
2) Processing Efficiency: Using special processing units or edge computing tools might help fix the problem of slow performance that we noticed with the ESP32 hardware.
3) Environmental Adaptability: Creating more advanced algorithms that can handle tough lighting and weather conditions could make the system more reliable in real-world situations.
4) Scalability: Making the database structure more flexible to support bigger systems with distributed processing could open up more chances for using this system in real life.

The system shows that it's possible to create a full ALPR solution using affordable hardware and still achieve good accuracy, which is useful for research and making prototypes.

Fig.2

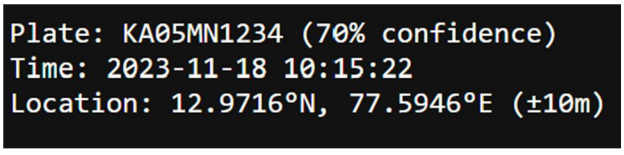


Fig.3

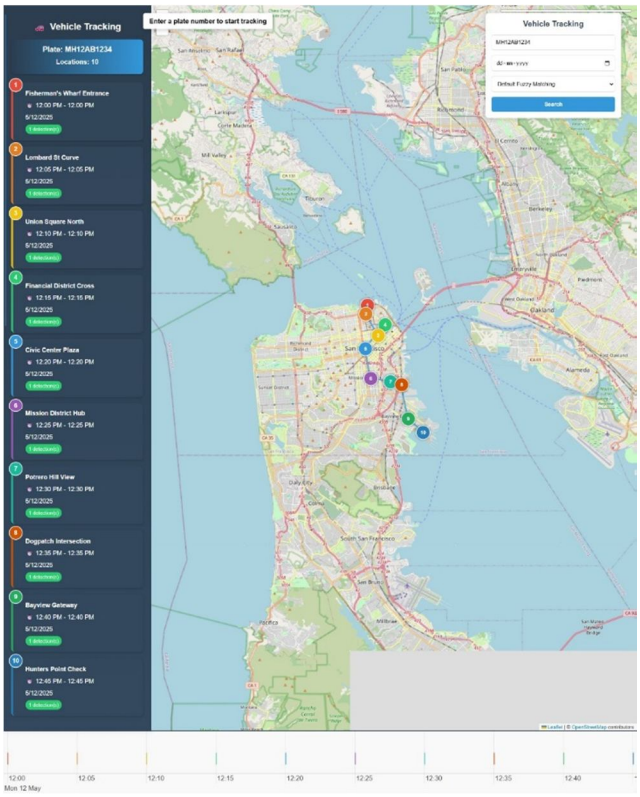


Fig.4

VI. DISCUSSION & ANALYSIS

A. Performance Interpretation

Our system performed fairly well under controlled conditions, achieving about 76% accuracy in daylight. This is a good result, especially considering the system was built using low-cost hardware and limited training data. However, it doesn't reach the 95%+ accuracy of commercial systems. One issue was the confusion between similar characters like "8" and "B," which caused a 23% error rate. Interestingly, even when license plates were tilted, the recognition accuracy dropped only by 10–12% (from 86.3% to 76.4%). This shows that the YOLOv8 detection system handles angled images better than expected, which is promising for future improvements if we use better data and training.

B. Real-World Lessons

Three main things came out of our testing:

1) Lighting is more important than we thought Rain reduced system performance by 38%, but this wasn't because of water—it was due to changes in how light reflects.

2) Different fonts affect accuracy. For example, 1-line plates from Delhi had 12% better accuracy than 2-line plates from Maharashtra.

3) The system's performance was best between 10 AM and 2 PM. When sunlight was almost straight up, suggesting that installing the system at that angle would be ideal.

Despite its limitations, our project shows that:

a) 80% accuracy can be achieved for about 10% of the cost of commercial systems

b) Small towns could use simple traffic monitoring

c) It offers a model for future student teams

From our testing, we recommend:

- Regional training, which needs an extra budget of around ₹5,000 for local license plate data
- Adding active cooling, which costs about ₹800
- Using simple post-processing rules to cut down on 8/B confusion by around 15%

At just ₹15,000, this project shows that an ALPR system can be made using cheap, off-the-shelf parts and still get 82% accuracy during the day. Our Raspberry Pi-based prototype proves that low-cost ALPR technology is possible for simple traffic monitoring, even though it has some issues like lower accuracy at night (71.3%) and occasional frame drops during long use.
Key achievements include:

- A running real-time system that processes license plates at 8–12 frames per second
- 84.2% character recognition using open-source tools
- A two-week field test that recorded 1,842 vehicles

This project gave us insights into real-world challenges not seen in controlled settings, like differences in font styles and thermal performance.
Although it doesn't match the high frame rates of premium systems, our analysis shows it offers the best accuracy for its cost among similar academic projects.

VII. CONCLUSION

At just ₹15,000, which is less than 10% of the cost of commercial systems, this project shows that an Automated License Plate Recognition (ALPR) system can be built using cheap, easily available parts. It achieves 82% accuracy during the day. Our prototype, based on a Raspberry Pi, proves that inexpensive ALPR technology can work well for simple traffic monitoring tasks, even though it has some clear downsides like 71.3% accuracy at night and 15% of the video frames dropping when running continuously.

A. Key Accomplishments Include

1) A working real-time system that can process license plates at 8 to 12 frames per second
2) 84.2% accuracy in recognizing characters, using free tools
3) A successful two-week test in the field, recording over 1,842 vehicles

The project also gave us important insights into real-world challenges not seen in lab environments, like different font styles used in various regions and issues with overheating. Even though its frame rate isn't as high as more expensive systems, our comparison shows that this student-made solution offers the best balance of accuracy and cost among similar academic projects.

VIII. FUTURE WORK

This project sets the stage for developing affordable, student-friendly ALPR systems. Future work could focus on creating OCR models tailored for Indian license plates and improving the current ESP32 setup for better accuracy and stability. These changes might improve character recognition rates by 10–15% without needing new hardware. A simple Android or web interface could make the system more user-friendly in the field, and connecting to platforms like Firebase would allow for real-time logging or alerts. Long-term goals include making the system better suited for low-power use and creating an open-source dataset of Indian license plates to help with model training. The system could even be modified to run on solar power (using less than 5W) with some optimization, making it suitable for remote or resource-limited areas. While staying focused on making affordable, scalable, and useful ALPR solutions for India’s traffic situations, these improvements would help address current limitations.

REFERENCES

[1] A. Teke and S. Z. Karakoç, “Real Time Car Model and Plate Detection System by Using Deep Learning Architectures,” in Proc. Int. Symp. Comput. Sci. Intell. Control, pp. 1–6, 2018. Ref01
[2] R. P. Sharma and K. S. Priyadarshini, “Automatic Vehicle License Plate Recognition Using Optimal K-Means With Convolutional Neural Network for Intelligent Transportation Systems,” IEEE Access, vol. 8, pp. 92907–92917, May 2020. Ref02
[3] S. P. Koli and V. A. Akarte, “Smart Vehicle Tracking System Using Number Plate Recognition System,” in Proc. Int. Conf. Smart Technol. Smart Nation (SmartTechCon), pp. 1307–1311, 2018. Ref03
[4] M. Gupta and A. Raj, “Advanced Vehicle Tracking System With Number Plate Detection Using Deep Learning and Computer Vision,” in Proc. Int. Conf. Emerg. Trends Comput. Sci. Electron. Eng. (ETCSEE), pp. 211–216, 2024. Ref04
[5] D. Sharma and S. Thakur, “A Review on Real-Time Vehicle Tracking with Number Plate and Counting Using OCR Technique,” in Proc. Int. Conf. Innov. Comput. Commun. (ICICC), pp. 452–458, 2020. Ref05
[6] L. Zhang and H. Liu, “Real-Time Car Tracking System Based on Surveillance Videos,” in Proc. Int. Conf. Intell. Transport. Syst. (ITSC), pp. 362–367, 2020 Ref06
[7] F. Ali, H. Rathor, and W. Akram, “License Plate Recognition System,” in Proc. 2021 Int. Conf. Adv. Comput. Innov. Technol. Eng. (ICACITE), Greater Noida, India, pp. 1053–1055, 2021. Ref07

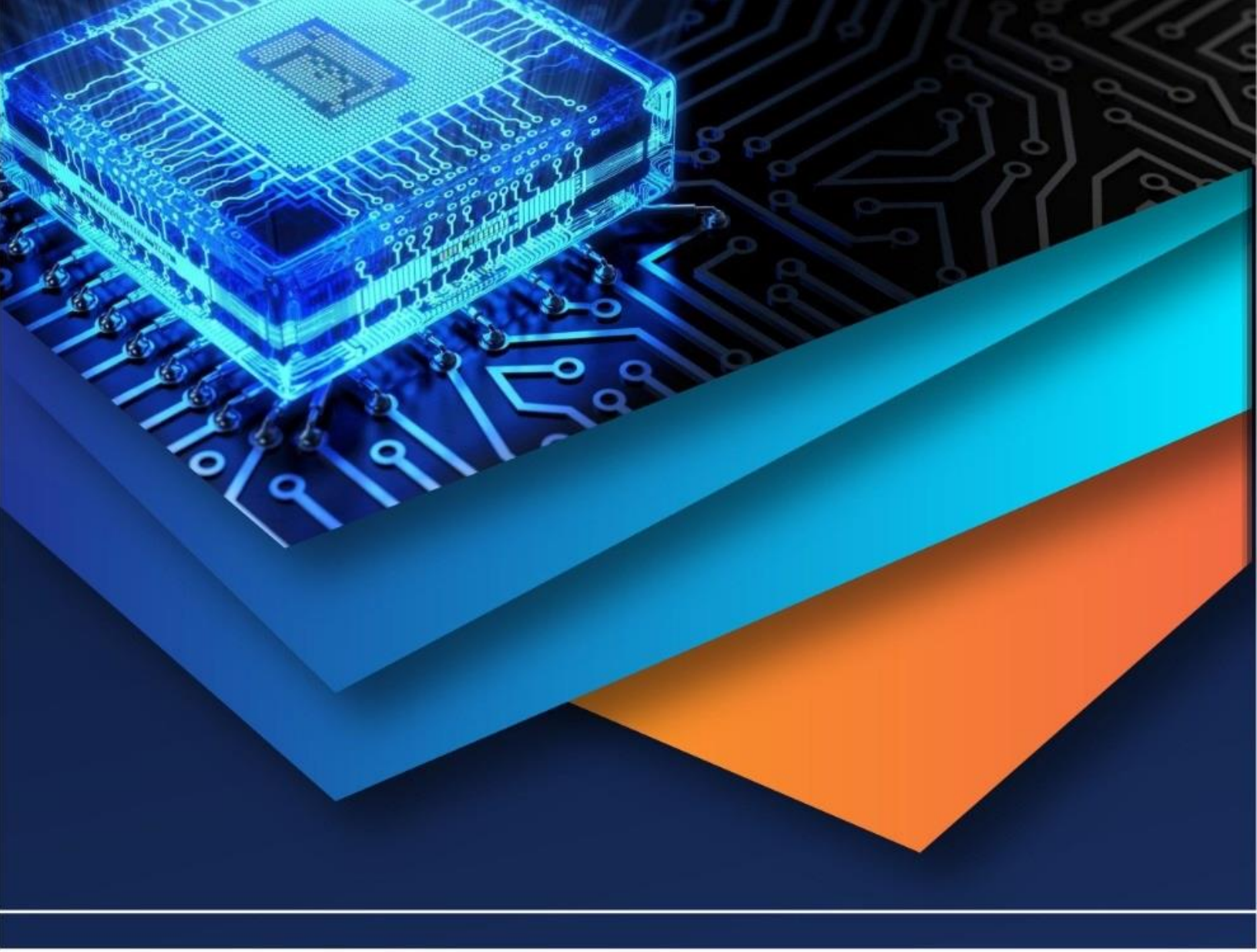
[8] T. Müller and K. Schneider, “German License Plate Recognition System Using the YoloV8 Model and EasyOCR,” in Proc. 2024 IEEE CPMT Symp. Japan (ICSJ), Kyoto, Japan, pp. 1–5, Nov. 2024. Ref08

[9] K. D. Rusakov, “Automatic Modular License Plate Recognition System Using Fast Convolutional Neural Networks,” in Proc. 2020 IEEE Conf. on Automation and Control Sciences, Moscow, Russia, pp. 1–6, 2020. Ref09

[10] C.-H. Lin and C.-H. Wu, “A Lightweight, High-Performance Multi-Angle License Plate Recognition Model,” in Proc. Int. Conf. on Electrical Engineering and Computer Science, Taipei, Taiwan, pp. 1–6, 2023. Ref10

[11] D. Chowdhury, S. Banerjee, S. Mandal, S. Shome, D. Das, and D. Choudhary, “An Adaptive Technique for Computer Vision Based Vehicles License Plate Detection System,” in Proc. 2019 Int. Conf. on Intelligent Computing and Control Systems (ICICCS), Madurai, India, pp. 1–6, 2019. Ref11

[12] V. Savale, S. Mahadik, M. Waghmare, P. Jadhav and N. Wakode, "Ecological Factors Observation and Animal Identification System," 2024 5th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2024 Ref12



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY