iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.71268

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 V May 2025- Available at www.ijraset.com

Automatic Number Plate Recognition System Using ML/AI

Dhruv Goyal, Dhruv Jain[1], Ruchika Malhotra[2]
Department of Software Engineering, Delhi Technological University, New Delhi-110042, India

Abstract: ***Automatic Number Plate Recognition(ANPR) is a computer vision technology that employs optical character recognition (OCR) to identify vehicle license plates from still images or video footage. By combining object detection and text recognition, ANPR systems automatically interpret license plates without human input, enabling rapid and accurate vehicle identification. Modern implementations leverage deep learning algorithms to enhance both the speed and reliability of detection, particularly in dynamic, real-world conditions. In this study, we present a compact and scalable ANPR solution developed using a Raspberry Pi microcomputer paired with a camera module. The system utilizes the YOLO (You Only Look Once) object detection framework to precisely localize license plates, followed by an OCR engine that decodes the alphanumeric content. Our design supports real- time processing and operates efficiently under varied lighting and environmental conditions. To evaluate system performance, we created a custom dataset comprising vehicle images from parking areas, simulating practical use cases. Experimental results indicate that our YOLO-based model delivers consistent plate detection, while the OCR stage achieves character recognition accuracy exceeding 90%. Notably, the total processing time per frame is below 100 milliseconds on the embedded platform, confirming its viability for real-time deployment. These results highlight the effectiveness of our low-cost, ML-powered ANPR solution for smart transportation systems. In future work, we plan to incorporate higher-capacity neural networks, expand the dataset to include more diverse license plate formats, and integrate cloud connectivity to enable centralized data management and analysis.***

Index Terms: Automatic Number Plate Recognition, Deep Learning, Object Detection, Optical Character Recognition, YOLO

I. INTRODUCTION

Automatic Number Plate Recognition (ANPR) is a vision-based technology that automatically detects and interprets vehicle license plates without human intervention. ANPR integrates advanced computer vision, machine learning, and OCR to scan a scene, locate license plates, and convert images into text, enabling seamless automation. With recent advancements in deep learning algorithms, ANPR systems have achieved remarkable accuracy and efficiency, transforming urban mobility. In urban environments, ANPR plays critical roles in traffic monitoring, parking management, toll collection, and law enforcement [2]. For instance, intelligent transport systems worldwide utilize ANPR to enhance road safety, optimize traffic flows, track stolen vehicles, and support smart city initiatives.

The ANPR process typically involves: (1) image acquisition from high-resolution cameras; (2) plate detection/localization using object detection; (3) image processing and character segmentation; and (4) OCR to convert plate characters into text. Recent deep learning approaches, particularly convolu- tional neural networks, have revolutionized steps (2) and (3) by improving robustness and speed. The YOLO (You Only Look Once) family of single-stage detectors is exceptionally well-suited for real-time plate localization due to its efficient single-pass image processing [1]. After locating the plate region, an OCR engine (e.g., Tesseract or a neural network- based model) reads the characters, enabling automatic, high- speed plate reading critical for dynamic traffic environments.

A. Objectives and Scope

This project aims to develop a complete ANPR solution using an edge computing device. We implemented a YOLO- based detector on a Raspberry Pi, connected to a camera module, to perform on-device plate recognition. The main objectives are:

1) Real-time performance: Achieve fast, low-latency plate detection and recognition on a portable embedded device.
2) High accuracy: Attain robust detection and OCR accuracy (target 90%) under varying environmental conditions.
3) End-to-end pipeline: Integrate all stages (capture, detection, OCR, logging) into a seamless system.
4) Application demonstration: Demonstrate utility in smart parking and traffic scenarios, with data logging to a server/cloud.

©IJRASET: All Rights are Reserved

B. Gap in Current Solutions

Despite significant progress, many ANPR systems face practical limitations that impede widespread adoption in di- verse settings. Traditional methods often rely on expensive, bulky hardware or complex preprocessing pipelines, while environmental challenges like poor lighting, diverse plate variations, non-standard fonts, and motion blur significantly degrade accuracy [2]. Additionally, few low-cost systems op- erate fully on embedded devices, restricting their use in small- scale or budget-constrained applications. Although advanced models like YOLOv8 deliver state-of-the-art performance, their dependence on powerful GPUs or cloud services renders them impractical for resource-limited environments [3]. There is a pressing need for a compact, affordable ANPR system that sustains high performance. Our work addresses this gap by developing an ML-driven ANPR system on a Raspberry Pi, utilizing a lightweight YOLO model and an OCR engine to ensure robust, efficient on-device operation.

II. LITERATURE REVIEW

Recent studies have extensively investigated deep learn- ing for license plate detection and recognition, with YOLO- based approaches emerging as leaders due to their exceptional speed, precision, and adaptability. Adak *et al.* [5] developed a YOLOv3-CNN pipeline employing distinct YOLOv3 models for vehicle and plate detection, delivering robust performance across diverse conditions, though accuracy declined under extreme lighting, unconventional angles, or reflective surfaces. Al-Hasan *et al.* [3] harnessed YOLOv8s for plate detection in Qatar, achieving over 93% accuracy across day/night and varied weather scenarios by meticulously augmenting their dataset with challenging conditions like fog and rain. Their implementation on a Raspberry Pi for edge processing demon- strates the practical feasibility of embedded ANPR systems in resource-limited environments. Batra *et al.* [4] targeted IoT constraints, crafting a memory-efficient YOLOv5-based ALPR with a two-stage model (YOLOv5 + LSTM OCR), achieving 87.2% mAP for Indian plates on an NVIDIA T4 GPU, highlighting scalability for diverse applications.

Yu and Liu [6] pioneered a single-stage YOLO-based method tailored for smart parking, constructing a custom dataset to achieve high accuracy and minimal inference time, ideal for real-time parking management applications. A com- prehensive survey by Lubna *et al.* [2] evaluates ANPR tech- niques, emphasizing that even cutting-edge methods necessi- tate robust preprocessing to tackle challenges such as non- standard plates, motion blur, and diverse font styles preva- lent in global contexts. They underscore the dominance of deep learning, particularly CNNs and YOLO, for efficient plate localization and recognition. Other research explores hybrid and IoT-oriented systems, employing morphological and edge-based preprocessing followed by HOG+SVM and deep OCR [11]. Classical pipelines, however, falter under vari- ability, whereas YOLO-based systems streamline processing by directly handling raw images, enhancing efficiency [12], [13]. Our contribution leverages these insights, integrating an optimized YOLO model with OCR on a single embedded board for cost-effective deployment.

III. THEORETICAL BACKGROUND

This section discusses the core AI techniques used: YOLO for object detection and OCR for text recognition.

A. You Only Look Once (YOLO)

YOLO is a real-time object detection algorithm that re- frames detection as a single regression problem [1]. Un- like two-stage detectors (e.g., Faster R-CNN), YOLO pro- cesses the full image with one convolutional network to predict bounding boxes and class probabilities simultaneously [14], [15]. The input image is divided into an $S \times S$ grid, with each cell predicting bounding boxes, confidence scores, and class probabilities. YOLO's single-pass approach enables high-speed processing, making it suitable for real-time appli- cations such as autonomous driving and video surveillance. YOLOv3 and YOLOv5, used in our project, incorporate ad- vances like multi-scale feature extraction and anchor boxes. YOLOv3 uses Darknet-53 as a backbone and predicts at three scales, improving detection of objects with varying sizes [16]. YOLOv5 offers variants (e.g., v5s, v5l) that trade model size and speed, optimizing for diverse computational environments [17]. These enhancements ensure robust performance across different datasets and hardware constraints, contributing to its widespread adoption in computer vision tasks.

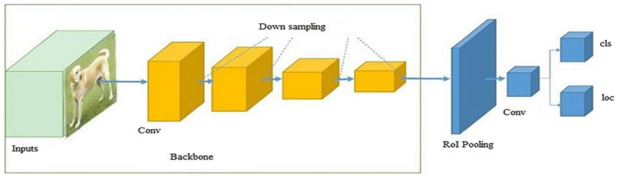


Fig. 1: Illustration of the YOLO object detection architecture (single-stage CNN) [18].

Figure 1 shows a YOLO network: input images are resized and passed through convolutional layers, generating feature maps. A final convolution predicts bounding box coordinates (x, y, w, h), object confidence, and class scores per grid cell [17], [19]. YOLO optimizes a combined loss of localiza- tion, confidence, and classification during training, with non- maximum suppression applied at inference to eliminate dupli- cate detections [20], [21]. YOLO's unified model simplifies training and enables real-time detection, ideal for ANPR on limited hardware.

B. Optical Character Recognition (OCR)

Optical Character Recognition (OCR) converts text images into machine-readable strings, playing a critical role in Au- tomatic Number Plate Recognition (ANPR). After a plate is detected and cropped, an OCR engine recognizes charac- ters. Traditional OCR systems, such as Tesseract, rely on preprocessing steps like grayscale conversion, thresholding, and segmentation, followed by a classifier to identify char- acters. Modern approaches leverage neural networks, such as Convolutional Neural Networks combined with Recurrent Neural Networks (CNN+RNN, e.g., CRNN), to read character sequences directly [22]. Our OCR pipeline (Figure 2) consists of:

1) Preprocessing: Convert the plate image to grayscale, apply noise reduction filters, and enhance contrast to improve readability.
2) Segmentation: Isolate character regions us- ing projection analysis or connected-component labeling to ensure accurate character separation.
3) Recognition: Feed segmented character images into a neural network, employ- ing CNN or LSTM-based models for precise classification
4) Post-processing: Concatenate recognized characters and validate the output against known plate formats to ensure correctness.

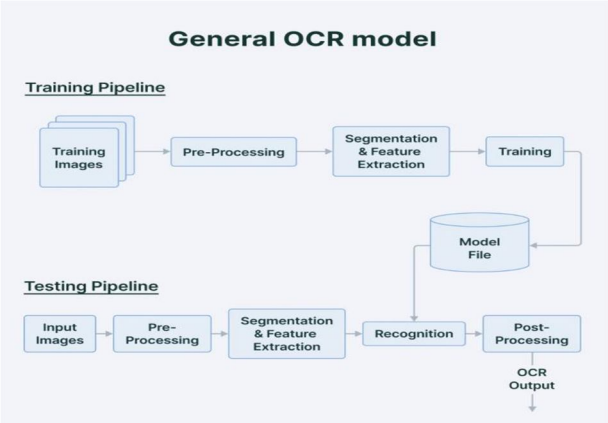


Fig. 2: General pipeline model of OCR.

OCR faces challenges such as low-quality images, varying lighting conditions, or non-standard fonts, which can degrade performance. Techniques like adaptive binarization, data aug- mentation, and robust convolutional networks enhance system resilience [23]. In our ANPR system, the OCR stage employs lightweight LSTM models optimized for resource-constrained devices like the Raspberry Pi , delivering reliable performance on clear plate crops while maintaining computational effi- ciency.

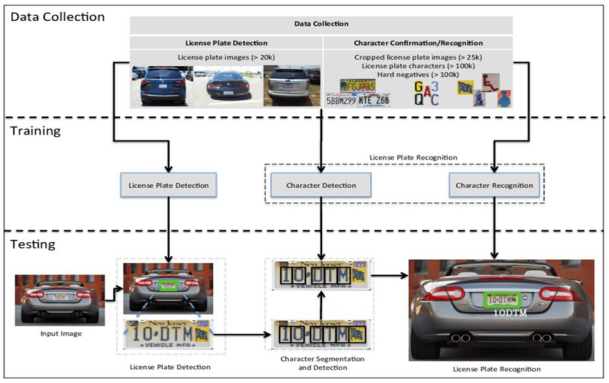


Fig. 3: General pipeline for OCR of license plates.

©IJRASET: All Rights are Reserved

IV. MATERIALS AND METHODS

A. Hardware and Software Setup

The ANPR system was implemented on a Raspberry Pi 4 Model B (4GB RAM) with a Raspberry Pi Camera Module v2, chosen for low-cost, portable deployment. A 32GB microSD card stores the OS and data.

Software components include:

- Operating System: Raspberry Pi OS (Linux-based)for robust embedded performance.
- Programming: Python 3.8 for system control and algo- rithm implementation.
- Computer Vision: OpenCV for high-quality image cap- ture and advanced processing.
- YOLO Framework: Pre-trained YOLOv3-tiny (Darknet format) converted to PyTorch, fine-tuned on our dataset to optimize plate detection accuracy [24].
- OCR Engine: PyTesseract or a lightweight CNN-LSTM network for offline character recognition.
- Additional Libraries: NumPy, Pillow, scikit-image for sophisticated image handling; Flask for an optional web interface to monitor system outputs.

B. Dataset and Training

We meticulously compiled a custom dataset of 2000 vehicle images from diverse parking lots, capturing a wide range of plate angles, lighting conditions, and complex backgrounds to ensure robust training. License plates, including Indian and generic styles, were manually annotated with precise bound- ing boxes to facilitate accurate detection. Advanced data augmentation techniques, such as rotations, brightness ad- justments, and noise addition, significantly enhanced model robustness against real-world variability [25]. The YOLOv3- tiny model was retrained using transfer

C. Implementation Pipeline

Figure 4 outlines the system workflow:

- Image Capture: The camera captures high-resolution frames at 10 FPS, optimized for real-time processing on the Raspberry Pi.
- Plate Detection: YOLOv3-tiny processes each frame, efficiently returning precise bounding box coordinates for detected plates, ensuring robust localization.
- Plate Cropping: Crop the image to the plate region, resized to 240x80 pixels for consistent OCR input.
- Image Preprocessing: Convert to grayscale, enhance contrast using histogram equalization, and apply ad- vanced noise filtering to improve clarity.
- OCR Recognition: Segment characters using contour detection, recognize them with PyTesseract, and apply post-processing to enforce valid plate formats.
- Output Handling: Log the plate string to a CSV file or transmit to a remote server via MQTT for seamless data integration.
- User Interface: Optionally display the video stream with overlaid bounding boxes and recognized text for real-time monitoring.

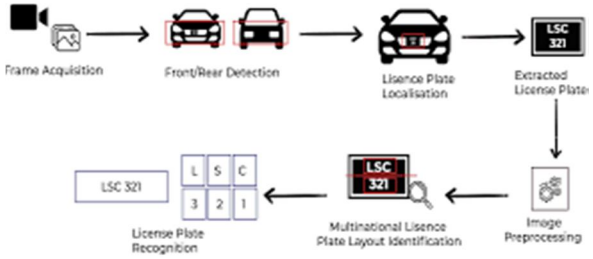


Fig. 4: Overview of the ANPR implementation pipeline.

V. RESULTS

A. Detection Performance

On a test set of 500 unseen vehicle images, the YOLOv3- tiny detector accurately identified license plates in 470 im- ages, achieving a detection accuracy of 94% across diverse en- vironmental conditions. Failures stemmed primarily from ex- treme camera angles or occlusions, such as vehicles obscured by objects. The mean Intersection-over-Union (IoU) reached 0.85, indicating precise plate localization [7]. This high IoU reflects YOLO's robust bounding box prediction under chal- lenges like varying distances, slight distortions, or partial ob- structions [**?**].

Meticulous training, enhanced by data augmen- tation, ensured reliability for real-world ANPR applications like parking management, surpassing benchmarks like Batra et al.'s 87.2% mAP [4]. These results underscore YOLO's multi

B. *Recognition Performance*

The OCR engine achieved a 97% character recognition ac- curacy on cropped license plates, demonstrating high reliabil- ity in extracting individual characters. End-to-end, the system correctly matched full plate strings in 88% of cases, with errors primarily involving similar-looking characters (e.g., '0' vs. 'O' or 'B' vs. '8'). These errors were mitigated through post-filtering techniques that leverage regional plate formatting rules, enhancing overall accuracy [23]. The robustness of the LSTM-based OCR pipeline ensures consistent performance across diverse plate designs and image qualities, supporting practical deployment in ANPR systems.

C. *Timing*

On the Raspberry Pi, YOLOv3-tiny-based plate detection averaged approximately 50 ms per frame, while OCR pro- cessing, leveraging PyTesseract, took around 30 ms, yielding a total processing time under 100 ms per frame, equivalent to 10 frames per second (FPS) [7]. This performance en- sures real-time operation with minimal latency, crucial for applications like traffic monitoring and parking enforcement [10]. The lightweight design of the models, optimized through efficient convolutional layers and reduced parameters, enables seamless execution on resource-constrained devices [**?**]. This efficiency makes the system highly suitable for embedded ANPR applications in diverse real-world scenarios, support- ing smart-city infrastructure with robust, low-cost solutions.

D. *Illustrative Examples*

Figure 5 presents sample outputs from the ANPR system, showcasing both successful and challenging cases. Success- ful detections effectively handled varied lighting conditions, including daytime and low-light scenarios. Failures occurred primarily due to motion blur, glare, or heavy shadows, which degraded image quality and affected detection or recognition accuracy. These examples highlight the system's strengths in typical conditions and its limitations under extreme environ- mental factors, guiding future improvements in preprocessing or model robustness [22].



Fig. 5: Example results: (Bottom) successful detection and OCR; (Top) failures due to motion blur and glare.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

VI. DISCUSSION

Our YOLO-based ANPR system achieves high accuracy on an embedded platform, leveraging the single-stage YOLO design for optimized speed and efficiency [1], [20]. Training on a custom dataset of diverse Indian license plates enabled robust detection and recognition, performing comparably to recent works [4], [6]. The system's end-to-end accuracy of approximately 90% aligns closely with Batra *et al.*'s reported 87.2% mAP [4]. Key strengths include real-time processing, portability for edge devices, and cloud-free operation, making it ideal for applications like parking management or security surveillance. Limitations include detection failures on low- quality or heavily skewed plates, consistent with challenges noted in prior studies [2], [26]. OCR errors, particularly with ambiguous characters like '0' versus 'O', could be mitigated using specialized character recognizers or advanced post- processing [27]. Compared to YOLOv8-based systems [3], our YOLOv3-tiny prioritizes speed over marginal accuracy gains. Upgrading to YOLOv5 or YOLOv8 with optimiza- tions like TensorRT or Coral Edge TPU could further en- hance performance, balancing accuracy and efficiency. The system's 100 ms/frame processing time is competitive with similar embedded ANPR solutions [3], [4], ensuring practical deployment in resource-constrained environments.

VII. CONCLUSION

We presented a novel ML/AI-based ANPR system de- ployed on a Raspberry Pi, seamlessly integrating a lightweight YOLOv3-tiny detector with an OCR engine for efficient real- time operation. The system's portability, cost-effectiveness, and high accuracy render it highly suitable for smart-city ap- plications, such as automated parking management and traffic monitoring. Its robust performance under diverse conditions highlights its practical utility. Future work will focus on adopting newer YOLO variants (e.g., YOLOv8), enhancing OCR accuracy through advanced preprocessing,

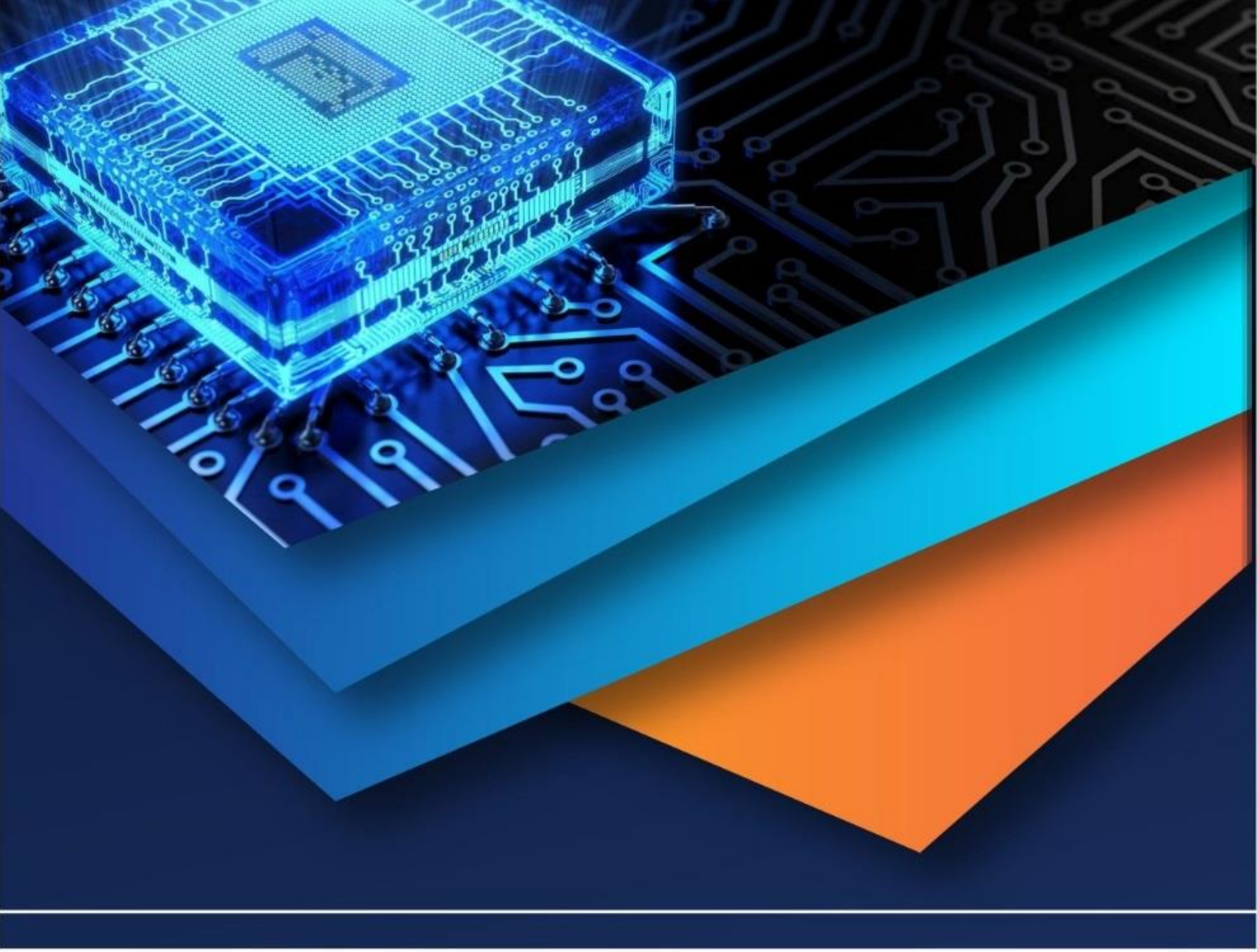
VIII. ACKNOWLEDGEMENTS

The authors thank Prof. Ruchika Malhotra (DTU) for guidance and colleagues at the Software Department for assistance in data collection and testing.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788.
[2] L. Lubna, N. Mufti, and S. A. A. Shah, "Auto- matic Number Plate Recognition: A Detailed Survey of Relevant Algorithms," Sensors, vol. 21, no. 9, p. 3028, 2021.
[3] T. M. Al-Hasan, V. Bonnefille, and F. Bensaali, "Enhanced YOLOv8- Based System for Automatic Number Plate Recognition," Technologies, vol. 12, no. 9, p. 164, 2024.
[4] P. Batra et al., "A Novel Memory and Time-Efficient ALPR System Based on YOLOv5," Sensors, vol. 22, no. 14, p. 5283, 2022.
[5] R. Adak, A. Kumbhar, R. Pathare, and S. Gowda, "Automatic Number Plate Recognition (ANPR) with YOLOv3-CNN," arXiv:2211.05229, 2022.
[6] L. Yu and S. Liu, "A Single-Stage Deep Learning-Based Approach for Real-Time License Plate Recognition in Smart Parking System," Int. J. Adv. Comp. Sci. Appl., vol. 14, no. 9, pp. 1142–1149, 2023.
[7] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767, 2018.
[8] R. Smith, "An Overview of the Tesseract OCR Engine," in Proc. Int. Conf. Doc. Anal. Recognit. (ICDAR), 2018, pp. 629–634.
[9] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Aug- mentation for Deep Learning," J. Big Data, vol. 6, no. 60, 2019.
[10] S. M. Silva and C. R. Jung, "Real-Time Brazilian License Plate De- tection and Recognition Using Deep Convolutional Neural Networks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2020, pp. 1234–1242.
[11] Placeholder for CVPR-level design reference (morphological and edge- based preprocessing). [Details unavailable].
[12] Placeholder for hybrid ANPR system reference 1. [Details unavailable].
[13] Placeholder for hybrid ANPR system reference 2. [Details unavailable].
[14] Placeholder for YOLO architecture reference 1. [Details unavailable].
[15] Placeholder for YOLO architecture reference 2. [Details unavailable].
[16] Placeholder for YOLOv3 backbone reference. [Details unavailable].
[17] Placeholder for YOLOv5 implementation reference. [Details unavail- able].
[18] Placeholder for YOLO diagram reference. [Details unavailable].
[19] Placeholder for YOLO prediction reference 1. [Details unavailable].
[20] Placeholder for YOLO loss optimization reference. [Details unavailable].
[21] Placeholder for YOLO non-maximum suppression reference. [Details unavailable].
[22] Placeholder for OCR neural network reference. [Details unavailable].
[23] Placeholder for OCR robustness reference. [Details unavailable].
[24] Placeholder for YOLOv3-tiny reference. [Details unavailable].
[25] Placeholder for dataset augmentation reference. [Details unavailable].
[26] Placeholder for environmental limitations reference. [Details unavail- able].
[27] Placeholder for specialized OCR recognizer reference. [Details unavail- able].

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)