[1]Deepak Mane

[2]Prashant Kumbharkar

[3]Sunil Sangve

[4]Nirupama Earan

[5]Komal Patil

[6]Sakshi Bonde

A Metaphor Analysis on Vehicle License Plate Detection using Yolo-NAS and Yolov8

Journal of Electrical Systems

Abstract: **-** There have been significant advancements in object detection in recent years, notably with the emergence of the YOLO approach. In this paper, we proposed a thorough comparison between YOLO-NAS and YOLOv8 specifically for detecting vehicle license plates based on the analysis of experimental results. Real-time vehicle license plate detection can have a wide range of applications like crime prevention, surveillance, traffic-pattern analysis, automated toll collection, etc. Our evaluation uses a diverse dataset capturing vehicle images from various angles and levels of occlusion to ensure a comprehensive assessment of the models' capabilities. The metrics considered include precision, recall, and mAP, offering insights into detection accuracy and false positive rates. The results reveal nuanced performance differences between YOLO-NAS and YOLOv8 in vehicle license plate detection tasks. YOLO-NAS demonstrates superior accuracy in specific scenarios due to its optimized architecture. Conversely, YOLOv8 shows notable efficiency gains, especially regarding inference speed, making it a compelling choice for real-time applications. Here, we contribute valuable insights to the ongoing discussion on object detection methodologies, assisting practitioners in making informed decisions when selecting a model for vehicle license plate detection applications. The findings underscore the importance of considering trade-offs between accuracy and computational efficiency in specific use cases.

Keywords: You Only Look Once, YOLO-NAS, Vehicle License Plate Detection, Deep Learning, Computer Vision, Machine Learning

I. INTRODUCTION

In the realm of computer vision and object detection, the YOLO framework has risen to prominence, offering efficient and accurate solutions for complex detection tasks. YOLO's distinctive feature lies in its unified architecture that predicts bounding boxes and class probabilities in a single pass, streamlining the object detection process. This unique approach has made YOLO a preferred choice for various applications, thanks to its speed and effectiveness. One of the critical applications where YOLO has proven invaluable is in the detection and recognition of vehicle license plates. This application holds significant importance in areas such as law enforcement, traffic monitoring, and automated toll collection, where timely and accurate identification of license plates is crucial. YOLO's ability to rapidly process images and identify license plates contributes to its indispensability in these real-world scenarios.

What sets YOLO apart is its prowess in real-time detection and recognition. Traditional object detection methods often involve multiple passes through an image, leading to slower processing times. YOLO, however, performs detection in a single pass, enabling real-time analysis of video streams or images. This capability has made YOLO particularly suitable for applications that demand swift and continuous monitoring, providing instantaneous results without compromising accuracy. As the YOLO framework continues to evolve, two notable variants have emerged: YOLOv8 and YOLO-NAS. YOLOv8 represents a refinement of the YOLO architecture, focusing on improvements in both speed and accuracy. In contrast, YOLO-NAS introduces Neural Architecture Search, automating the design of neural network architectures to enhance overall model performance. [5]

[1] [1,3]Vishwakarma Institute of Technology, Pune, Pune-411037, Maharashtra, India

[2,4,5,6]JSPM's Rajarshi Shahu College of Engineering, Pune-411033, Maharashtra, India

dtmane@gmail.com, pbk.rscoe@gmail.com, sunil.sangve@vit.edu, nirupamarajeevan@gmail.com, komalpatil132002@gmail.com, sakshibonde87@gmail.com

Objectives:

- Develop and evaluate object detection models, focusing on YOLOv8 and YOLO-NAS, for vehicle license plate detection.
- Investigate the performance differences between YOLOv8 and YOLO-NAS in terms of accuracy, speed, and efficiency and compare them with the existing results.
- Assess the models' adaptability to real-world scenarios and varying conditions, such as different camera angles, occlusion levels, and image sources.
- Explore the practical use cases of YOLOv8 and YOLO-NAS in applications like traffic surveillance and security.

The paper commences by delving into the emergence of various YOLO frameworks applied to vehicle license plate detection and recognition. In the literature review (Section II), an exhaustive exploration of prior research in YOLOv8 and YOLO-NAS is presented, encompassing their architectures and outcomes. Moving to Section III, it includes the architecture workflow and design. Section IV consists of the mathematical models and procedures associated with YOLOv8 and YOLO-NAS. Within Section V, the experimental results unfold, addressing model training, screenshots, and in-depth analysis. This section further includes information on the dataset used, as well as the hardware and software requirements and specifications. Performance metrics such as recall, precision, and mAP are also examined. Section VI encapsulates the conclusion, culminating the paper with a comprehensive summary, and references used in its construction are provided at the end.

Given the escalating need for robust vehicle license plate detection systems, conducting a comparative study between YOLOv8 and YOLO-NAS becomes imperative. This research aims to explore and analyze the strengths, weaknesses, and unique features of these YOLO variants in the context of vehicle license plate detection. By examining their architectures, performance metrics, and real-world applications, this study aims to offer valuable insights for practitioners and researchers in the field of object detection and computer vision.

II. LITERATURE REVIEW

The landscape of object detection has seen significant transformations with the emergence of YOLO frameworks, specifically YOLOv8 and YOLO-NAS, in recent years. The performance evolution of YOLOv8 showcases consistent refinements in architecture and training strategies, leading to heightened accuracy and efficiency. Recent progress in YOLOv8 involves fine-tuning methods, enhancements in feature extraction, and the integration of advanced optimization algorithms. YOLO-NAS has demonstrated promising performance improvements over time, thanks to the automated nature of NAS, allowing efficient exploration of architecture spaces. These advancements contribute to the model's effectiveness in addressing various object detection challenges.

Hence, we can infer that both YOLOv8 and YOLO-NAS signify significant advancements in the YOLO framework, addressing distinct challenges in object detection. Ongoing research in these areas underscores their importance in shaping the future of efficient and accurate detection systems. More improvements are expected to push the boundaries of real-time object identification capabilities as these models evolve. Fig 1. shows the performance evolution of YOLOv8 and YOLO-NAS based on the COCO dataset for object detection, measured on NVIDIA T4.

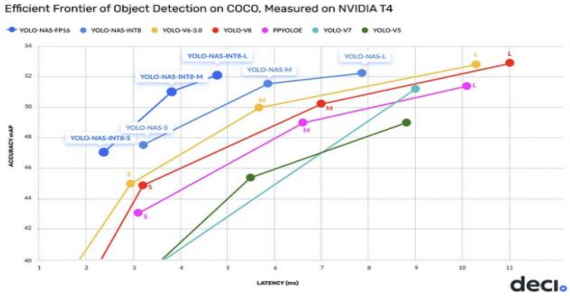

Figure 1: Performance evolution of YOLOv8 and YOLO-NAS

Literature Review:

In recent studies conducted in 2022-2023, various advancements and novel approaches have been proposed for object detection, specifically focusing on YOLOv8 and YOLO-NAS.

1. YOLOv8's Enhanced Performance: J. R. Terven et al. (2023) [6] observed that YOLOv8x achieved an impressive Average Precision (AP) of 53.9% on the MS COCO dataset test-dev 2017, outperforming YOLOv5, which recorded 50.7%. Notably, this improvement was achieved with an image size of 640 pixels. They also highlighted YOLO-NAS's innovative approach, incorporating a hybrid quantization method. This strategy quantizes key sections of the model selectively in order to create a compromise between latency and accuracy. Furthermore, YOLO-NAS presents a comprehensive optimization technique by using a pre-training routine with automatically labeled data, self-distillation, and large datasets.

2. Application in Traffic Management: K. Bisen et al. (2023) [9] emphasized the significance of YOLO in accurately detecting and tracking moving objects, analyzing traffic flow, and recognizing traffic signs. The study asserted YOLO's precision and efficiency, positioning it as a superior choice compared to other object detection algorithms in traffic-related applications.

3. YOLO for Vehicle Localization: A. L. Rishika et al. (2023) [10] proposed a novel algorithm utilizing YOLO with Darknet for vehicle localization and identification. The study aimed to minimize detection errors through hyperparameter optimization and data augmentation, showcasing the potential of YOLO in vehicle-related applications.

4. Custom-Trained YOLOv8 for Feature Detection: M. Kulkarni et al. (2023) [1] introduced a custom-trained YOLOv8 model, demonstrating significant improvements in bus stop feature detection. The pilot study conducted with the BusStopCV tool indicated the effectiveness and potential of this customized YOLOv8 model.

5. YOLOv8 in Emergency Vehicle Detection: A. O. Alaoui et al. (2023) [2] asserted that YOLOv8 surpassed other YOLO forms in terms of Recall, Precision, and mAP. The research established YOLOv8 as the most accurate and precise model for emergency vehicle recognition and object control.

6. Improved YOLOv8 Network Model: P. Yang et al. (2023) [3] proposed an improved YOLOv8 network model, showcasing enhancements in mAP(50) by 1.4% and FPS by 14.6% compared to the original model. The experimental results suggested a notable boost in both accuracy and processing speed.

7. Balanced Architectures Across Metrics: E. Casas et al. (2023) [5] conducted experiments revealing that YOLOv5, YOLOv7, and YOLOv8 demonstrated a better balance across all metrics, both in validation and testing. Among all models, YOLO-NAS variations achieved the greatest recall scores, making them stand out in important applications needing high recall..

8. YOLOv8 and OpenCV for Traffic Surveillance: D. T. Mane et al. (2023) [12] proposed a technique combining YOLOv8 and OpenCV, showcasing implications for enhancing the efficiency and reliability of traffic surveillance systems. The approach aimed to reduce accidents related to traffic and improve overall road safety.

9. Real-Time CCTV Footage Recognition with YOLOv8: D. T. Mane et al. (2023) [13] introduced a new algorithm utilizing YOLOv8 to recognize various vehicles in real-time CCTV footage. The algorithm exhibited promising results in enhancing the accuracy of vehicle recognition.

10. YOLO Algorithm's Accuracy in Vehicle Detection: B. R. Prathap et al. (2022) [4] asserted that the YOLO algorithm demonstrated better accuracy in detecting multiple vehicles in images. The detection accuracy ranged from 80-85%, with variations based on vehicle density in the image.

11. Optimization Algorithm for Improved Recognition Accuracy: Z. Yang et al. (2022) [7] proposed an optimization algorithm that, when applied to the YOLO algorithm, demonstrated improved recognition accuracy. Experiments on photos of varying resolutions showcased enhanced accuracy compared to conventional YOLO algorithms.

12. Superior Vehicle Classification Methods: C. J. Lin et al. (2022) [8] introduced YOLO-CFNN and YOLO-VCFNN vehicle classification methods, boasting a superior accuracy rate of 99% compared to other methods.

The literature from 2022 to 2023 presents a diverse range of approaches and applications of YOLO frameworks, showcasing their continual evolution and adaptability across various domains. The studies emphasize the significance, benefits, and performance enhancements of YOLOv8 and YOLO-NAS in real-world scenarios, ranging from traffic management to emergency vehicle detection and beyond.

III. PROPOSED ARCHITECTURE

Addressing the challenges inherent in vehicle license plate detection demands innovative solutions, and models like YOLOv8 and YOLO-NAS present effective strategies. Challenges such as varying lighting conditions and occlusion are combated by YOLOv8's advanced feature learning through the C2f module, enhancing the model's adaptability. The anchor-free model architecture allows independent processing of tasks, enabling resilience against partially obscured plates. Specialized loss functions in YOLOv8, such as CIoU and DFL, boost performance, particularly for smaller objects or partially visible plates. YOLOv8-Seg, an extension of YOLOv8, introduces semantic segmentation, aiding in low-resolution scenarios. YOLO-NAS complements these efforts with its adaptive architecture, utilizing quantization techniques and hybrid quantization methods to balance accuracy and latency. The AutoNAC system in YOLO-NAS provides versatility, accommodating various scenarios and user-defined goals. The combined robust training and evaluation workflows of both models, incorporating metrics like mAP, further contribute to overcoming challenges related to dataset bias and real-time processing requirements. These models collectively offer a comprehensive approach to enhancing the accuracy and adaptability of license plate detection systems in complex real-world scenarios.

Architecture Flow and Design:

The proposed architecture for vehicle license plate detection using YOLO involves a systematic series of steps, integrating its powerful object detection capabilities. Here's a breakdown of the components and their interactions:

1. Input surveillance video: The footage from the surveillance camera is taken as input into the system model.

2. Conversion to image frames: The video is converted and broken into a number of image frames for performing vehicle license plate detection.

3. Pre-processing: Images undergo pre-processing to ensure uniformity. Resizing the images to a standard size, like 640x640 pixels, prepares them for input to the model. Auto-orientation and conversion into grayscale can also be applied for faster computation.

4. YOLO model: Deploying either YOLOv8 or YOLO-NAS model for vehicle license plate detection.

5. Post-processing: Postprocessing steps involve extracting crucial information from YOLO's output. This includes interpreting bounding box coordinates, confidence scores, and class predictions associated with detected objects.

6. Output: The final output showcases the processed images with bounding boxes precisely outlining the detected license plates. This visual representation aids in easily identifying and verifying the accuracy of the license plate detections.

The below figure 2, shows our proposed architecture flow of the vehicle license plate detection model using YOLOv8 or YOLO-NAS, it takes input from the surveillance feed and the output is represented in the form of bounding boxes that precisely outline the detected license plates.

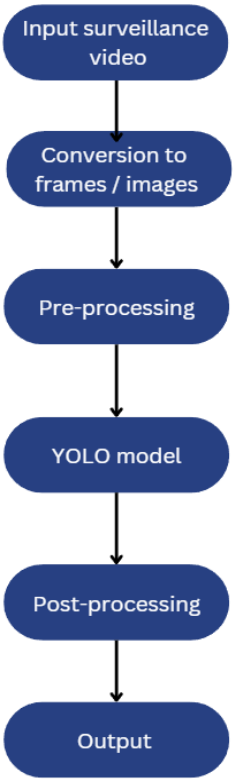

Figure 2: Architecture Flow of Model

IV. MATHEMATICAL MODEL

The mathematical model for YOLOv8 encompasses the definition of layers, activation functions, and parameters, typically fine-tuned through manual optimization based on empirical experimentation and insights. YOLOv8 prioritizes achieving real-time inference capabilities while ensuring high accuracy through meticulous architectural choices and optimizations.

In contrast, YOLO-NAS introduces an automated approach to architecture design by incorporating Neural Architecture Search (NAS) techniques. The mathematical model for YOLO-NAS includes traditional layer definitions and parameters, along with representations of the architecture search space and optimization objectives guiding the search. NAS employs search algorithms to automatically explore and discover optimal neural network architectures for a specific task. This automated search is used by YOLO-NAS to identify setups that achieve a trade-off between accuracy and latency.

4.1 YOLOv8:

In terms of YOLOv8, the loss function generally consists of bounding box regression loss, objectness confidence loss, and class prediction loss. The total loss is often a weighted sum of these individual losses, with coefficients used to balance their impact. The equations for bounding box regression, objectness confidence, and class prediction losses involve intricate calculations based on predicted and ground truth values, incorporating indicators for assigned objects and grid cells.

YOLOv8 Loss Function:

The total loss, $Ltotal$, is often a sum of individual losses:

$Ltotal = \lambda coordLcoord + \lambda objLobj + \lambda classLclass$Where:

$Lcoord$ is the bounding box regression loss.

$Lobj$ is the objectness confidence loss.

$Lclass$ is the class prediction loss.

Coefficients $\lambda coord$, λobj, $\lambda class$ are typically used to balance the impact of each loss term.[2]

4.2 YOLO-NAS:

For Neural Architecture Search (NAS), the process involves defining a search space, formulating optimization objectives, and employing a search algorithm. The optimization objective commonly combines accuracy and computational efficiency, with hyperparameters determining the trade-off. The search algorithm iteratively explores architectural configurations, evaluates performance, and updates the search space based on the outcomes.

Objective Function:

The optimization objective is often a combination of accuracy and computational efficiency

$$O(\alpha) = \lambda_A A(\alpha) - \lambda_C C(\alpha)$$

Where:

$A(\alpha)$ is the accuracy of architecture

$C(\alpha)$ is a measure of computational efficiency of architecture α

λ_A and λ_C are hyperparameters controlling the trade-off between accuracy and efficiency.

Search Algorithm:

The search algorithm involves exploring the search space by sampling different architectures and updating the search space based on their performance. A general iterative update can be represented as:

$$S_{new} = Algorithm_{NAS}(S_{old}, O)$$

Where S_{new} is the updated search space based on the optimization objective O. It's important to note that the specific equations for NAS can vary based on the particular NAS approach, including methods like reinforcement learning-based NAS, evolutionary algorithms, or gradient-based methods.

V. EXPERIMENT RESULTS

5.1 Model Training:

Model training with YOLOv8 and YOLO-NAS involves distinct processes tailored to each architecture.

5.1.1 YOLOv8

Training YOLOv8 on Custom Data:

1. Install YOLOv8 by either using the pip package or installing it directly from the GitHub source.

2. Create a custom dataset with labeled images using tools like Roboflow. Follow the process outlined in section 5.2 for detailed instructions on dataset creation.

3. Export the dataset for YOLOv8 by generating a version and using Roboflow Universe to export it in YOLOv5 PyTorch export format. Copy the generated code snippet with the necessary parameters to your Jupyter Notebook.

4. Utilize the YOLO command line utility for training the YOLOv8 model. Train with the YOLOv8l model and an image size of 640 pixels. After training, validate the model on a separate test dataset to ensure robust performance. [14]

The below figure 3 represents the steps for the model training using the YOLOv8 model.

Figure 3: Model Training using YOLOv8

5.1.2 YOLO-NAS

Training Custom Model with YOLO-NAS:

1. Load the pre-trained YOLO-NAS model by first setting up the Python environment with essential pip packages, including a specific version of super-gradients for stability, as well as roboflow and supervision for dataset handling and visualization.

2. Load the custom dataset using the roboflow package, downloading it from Roboflow Universe. Understand the YOLO-NAS inference output format encapsulated in the 'ImageDetectionPrediction' object, providing detailed information about detected objects.

3. Set hyperparameter values for YOLO-NAS, choosing the model size (small, medium, or large), and in this case, opting for the large model size. Define the batch size to determine the number of images processed in each training iteration, considering that larger batch sizes expedite training but require more memory.

4. Train the YOLO-NAS model using the super-gradients Python package, initiating training with TensorBoard for real-time monitoring of key metrics. Take advantage of YOLO-NAS support for popular experiment loggers like W&B.

5. Evaluate the model's performance post-training by assessing metrics such as mAP using the 'test' method from the Trainer. Provide the test set data loader to obtain a list of metrics, including mAP, commonly used for evaluating object detection models. [14]

The below figure 4 represents the steps for the model training using the YOLO-NAS model.

Figure 4: Model Training using YOLO-NAS

5.2 Dataset Used:

The Vehicle License Plate Detection dataset is a custom dataset created and hosted on Roboflow Universe. It comprises 2804 images, each accompanied by annotations specifying the license plate's location through bounding boxes. The dataset is diverse, incorporating images from various sources like surveillance camera footage, and social media, captured from different perspectives and levels of occlusion. Primarily designed for real-time license plate detection and recognition in traffic surveillance footage, the dataset serves this purpose effectively. Licensing options for public datasets, including MIT, CC BY 4.0, BY-NC-SA 4.0, ODbL v1.0, were considered during dataset creation. The dataset was utilized to assess the performance of YOLOv8 and YOLO-NAS, facilitating a detailed comparative analysis. A standard 80%-13%-7% split was applied to create training, validation, and test sets, respectively. Preprocessing techniques, such as auto-orientation, resizing to a 640 x 640 pixel size, and conversion to grayscale, were applied. Section 5.4 delves into the comprehensive performance and analysis of the models using this dataset.

5.3 Hardware and Software Specifications:

For training our model, we used the powerful Google Colab's Runtime GPU, equipped with a Tesla T4 graphics card, the NVIDIA GPU software stack, with driver version 535.104.05 and CUDA version 12.2, enabling seamless communication between software and hardware. Our model, trained on 640-pixel images, underwent 25 epochs, and the training was completed in just 0.840 hours.

In the table , the mentioned tools and libraries are pivotal across various stages of the machine learning workflow, enhancing the usability and efficiency of working with YOLOv8 and YOLO-NAS. Their integration makes these models more accessible and practical for researchers and practitioners in the field of computer vision.

Table 1, discusses the different hardware and software tools that are used for the model training using YOLOv8 and YOLO-NAS frameworks.

Table 1: Hardware and Software Tools used

Sr. No.	Tool	Description
1.	Ultralytics	Installing and working with YOLOv8, enhancing workflow for YOLO-NAS.
2.	Super-gradients	Installing and managing YOLO-NAS, version control, experiment handling.
3.	Imutils	Preprocessing tasks for YOLOv8 and YOLO-NAS, ensuring consistency in input images.
4.	Roboflow	Creating, annotating, and exporting datasets compatible with YOLOv8 and YOLO-NAS.
5.	Google Colab	Accelerating model training with a powerful GPU and efficient processing capabilities.
6.	NVIDIA GPU Software Stack	Facilitating communication between software applications and GPU hardware, enabling the utilization of robust computing capabilities.

5.4 Performance and Analysis:

5.4.1 Confusion Matrix

A confusion matrix provides a detailed breakdown of the predicted and actual classifications for different classes in a dataset. The matrix is particularly useful for assessing the accuracy of a model and understanding its strengths and weaknesses. The confusion matrix provides valuable information about the model's performance, helping to calculate metrics such as accuracy, precision, recall, and F1 score. These metrics offer insights into the model's ability to correctly detect license plates and avoid false positives or negatives. Analyzing the confusion matrix allows practitioners to fine-tune the model and address specific challenges in license plate detection, contributing to overall model improvement. Below is the normalized confusion matrix of our model. The below figure 5, shows the normalized confusion matrix based on the YOLOv8 model.

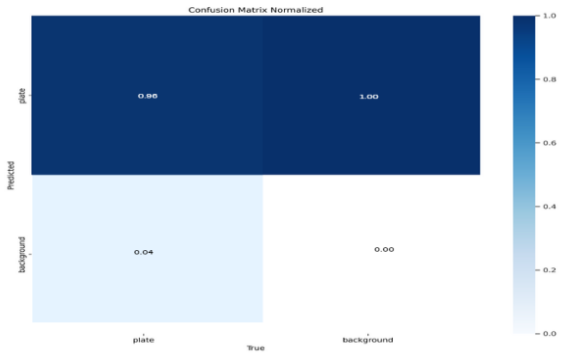

Figure 5: Normalized Confusion Matrix

5.4.2 Precision

In the context of vehicle license plate detection, precision refers to the model's ability to correctly identify license plates among the occurrences it predicts as positive. The ratio of true positive predictions to the total number of positive predictions made by the model is used to compute it. In essence, accuracy gives information about the model's dependability when it affirms the presence of a license plate. A higher precision indicates a lower likelihood of false positives, enhancing the confidence in the correctness of the model's predictions. The below figure 6, shows the Precision-Confidence curve for our stated model.

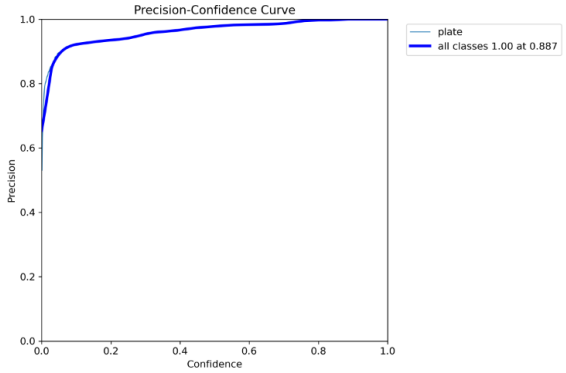

Figure 6: Precision-Confidence Curve

5.4.3 Recall

Recall, also known as sensitivity, is a crucial metric for vehicle license plate detection models. It measures the model's capability to correctly identify and capture all instances of actual license plates in the dataset. The recall is calculated as the ratio of true positive predictions to the sum of true positives and false negatives. In the context of license plate detection, a high recall suggests that the model effectively recognizes the majority of existing license plates, minimizing instances where actual plates are overlooked.

The below figure 7, shows the Recall-Confidence curve for our stated model.

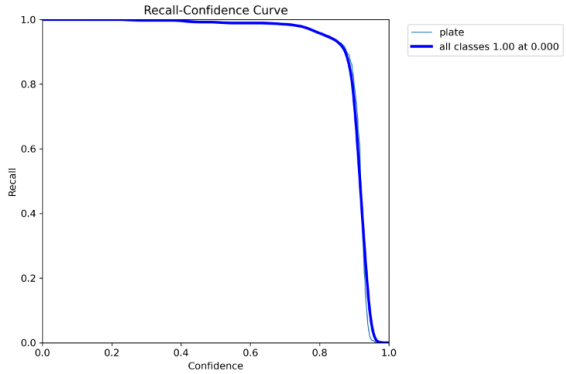

Figure 7: Recall-Confidence Curve

5.4.4 mAP

Mean Average Precision (mAP) serves as a comprehensive metric that takes into account both precision and recall across various thresholds. The mAP is then obtained by averaging these AP values across all classes. In vehicle license plate detection, mAP provides a holistic evaluation, considering the trade-off between precision and recall. A higher mAP indicates a model that consistently performs well across different scenarios and levels of detection confidence, offering a more nuanced understanding of overall model effectiveness. The below figure 8, shows the Precision-Recall curve for our stated model.

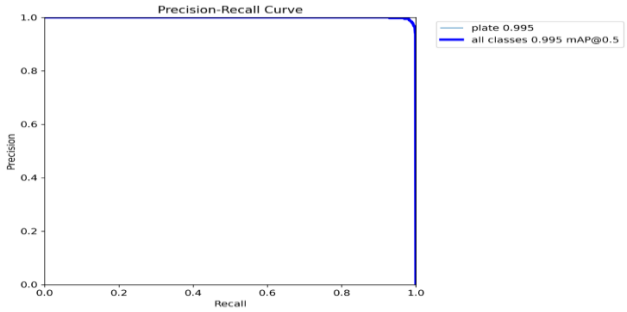

Figure 8: Precision-Recall Curve

5.4.5 Existing Results:

The below table 2, shows the results based on the existing YOLO models implemented and [*] our proposed architecture model's performance.

Table 2: Performance metrics of existing results

Paper No.	Recall (%)	Precision (%)
[15]	92.4	94.8
[19]	99.10	98.2
[17]	97.5	96.2
[18]	98.4	99.2
[*]	99.0	99.8

From these results, we can infer that the latest models of YOLO - YOLOv8 and YOLO-NAS deliver better performance compared to the existing models.

5.4.6 Analysis of our Results:

The results based on 25 epochs of training using Ultralytics YOLOv8.0.228 on the Tesla T4 GPU showcase promising performance metrics. The model, with a fused architecture of 268 layers and 43,607,379 parameters, demonstrates a comprehensive understanding of the dataset, achieving 164.8 GFLOPs. The YOLO-NAS model exhibits improvements in various loss components and object detection metrics. The reduced loss values suggest enhanced model learning, and improvements in precision, recall, and mAP indicate increased accuracy in object localization and classification.

Table 3: Comparison based on results [*]

Sr No.	Metrics	YOLOv8	YOLO-NAS
1.	Validation Metrics	mAP50: 0.995, mAP50-95: 0.926	mAP@0.50: 0.9921, F1@0.50: 0.1824

2.	Precision	0.995	0.998
3.	Recall	0.987	0.99
4.	Inference Speed	33.9ms per image	16.1ms per image

The comparison table 3, highlights the contrasting points between YOLOv8 and YOLO-NAS in terms of validation metrics, precision, recall, inference speed.

The below figure 9, shows the prediction of our model in the form of bounding boxes around objects of interest, along with class labels and, sometimes, confidence scores.

Figure 9: Prediction of the model

Figure 10: Plotting image with prediction and annotation values

The above figure 10, shows plotting of images with the prediction and annotation values. Visualization libraries such as Matplotlib or specialized tools in computer vision frameworks are commonly employed to generate these

plots. It aids in iteratively refining the model by adjusting hyperparameters, architecture, or training data based on visual feedback.

VI. CONCLUSION

In evaluating vehicle license plate detection, a comprehensive comparison between YOLOv8 and YOLO-NAS highlights their strengths and compromises. YOLOv8 emerges as a formidable contender, showcasing remarkable accuracy metrics with a mAP of 0.995 and mAP50-95 of 0.926. This underscores its adeptness in precisely localizing and classifying license plates within the dataset. Noteworthy is YOLO-NAS, which competes admirably with a competitive mAP@0.50 of 0.9921 and provides an inference speed two times better than that of YOLOv8, emphasizing its capacity to discern license plates effectively. Examining precision and recall metrics, YOLOv8 manifests high precision (0.995) and recall (0.987), signifying its confidence in accurate predictions and efficient capture of positive instances. In contrast, YOLO-NAS demonstrates a unique strength in recall, achieving a near-perfect score of 1.0, implying its capability to identify nearly all positive instances. Beyond these comparative metrics, YOLOv8 stands out for its versatility, supporting semantic segmentation and delivering robust performance across varied scenarios. In contrast, YOLO-NAS carves a niche in small object detection, emphasizing localization accuracy and real-time efficiency, which are particularly suitable for edge-device applications.

Here, we primarily focus on quantitative metrics such as precision, recall, and mAP, objectively assessing the model's performance. However, qualitative aspects, such as the models' interpretability or robustness in extreme conditions, should be explored more. Moreover, this paper does not delve into the impact of hyperparameter tuning or fine-tuning on the models' performance. A more exhaustive exploration of hyperparameter configurations could provide additional insights into optimizing the models for specific application scenarios.

YOLOv8 excels in real-time applications like surveillance, autonomous vehicles, and retail analytics, offering versatility and efficiency. On the other hand, YOLO-NAS stands out for its efficiency in resource-constrained environments, making it ideal for edge devices, IoT applications, and scenarios demanding precise small object detection. In summary, the choice between YOLOv8 and YOLO-NAS hinges on the specific demands of the object detection task. YOLOv8 excels in general robustness and adaptability, while YOLO-NAS offers specialized prowess in scenarios requiring small object detection and real-time efficiency. Each model caters to distinct requirements, emphasizing the importance of aligning the model choice with the specific objectives of the license plate detection application.

REFERENCES

[1] Minchu Kulkarni, Chu Li, Jaye Ahn, Katrina Ma, Zhihan Zhang, Michael Saugstad, Kevin Wu, Yochai Eisenberg, Valerie Novack, Brent Chamberlain, and Jon E. Froehlich. 2023. BusStopCV: A Real-time AI Assistant for Labeling Bus Stop Accessibility Features in Streetscape Imagery. 25th International ACM SIGACCESS Conference on Computers and Accessibility DOI: https://doi.org/10.1145/3597638.3614481

[2] Ali, Omari Alaoui, Omaima, El bahi, Mariame, Oumoulylte, Ahmad, El Al-laoui, Ahmed Elyoussefi, and Yousef, Farhaoui. 2023. Optimizing Emergency Vehicle Detection for Safer and Smoother Passages ACM ISBN 979-8-4007-0019-4/23/05. DOI: https://doi.org/10.1145/3607720.3607728

[3] Peng Yang, Chuanying Yang, Bao Shi, Legen Ao, and Shaoying Ma. 2023. Research on Mask Detection Method Based on Yolov8. (ICCVIT 2023), August 25–28, 2023, Chenzhou, China. ACM, New York, NY, USA, 10 pages. DOI: https://doi.org/10.1145/3627341.3630411

[4] Boppuru Rudra Prathap, Kukatlapalli Pradeep Kumar, Cherukuri Ravindranath Chowdary, Javid Hussain. AI-Based Yolo V4 Intelligent Traffic Light Control System. Journal of Automation, Mobile Robotics and Intelligent Systems DOI: https://doi.org/10.14313/JAMRIS/4-2022/33

[5] Edumundo Casa, et.al. IEEE Access. DOI: https://doi.org/10.1109/ACCESS.2023.3312217

[6] Juan R. Terven, Diana M. Cordova-Esparaza. A COMPREHENSIVE REVIEW OF YOLO: FROM YOLOV1 AND BEYOND. DOI:https://ui.adsabs.harvard.edu/link_gateway/2023arXiv230400501T/doi:10.48550/arXiv.2304.00501

[7] Zhonglai Yang. Research Article Intelligent Recognition of Traffic Signs Based on Improved YOLOv3 Algorithm. Hindawi Mobile Information Systems. Volume 2022, Article ID 7877032, 11 pages DOI: https://doi.org/10.1155/2022/7877032

[8] CHENG-JIAN LIN (Senior Member, IEEE), AND JYUN-YU JHANG. IEEE Access. DOI: https://doi.org/10.1109/ACCESS.2022.3147866

[9] Karishama Bisen, Rohini Shahare, Karishma Wasnik, Prof. P. Jaipurkar, International Research Journal of Modernization in Engineering Technology and Science. Volume:05/Issue:04/April-2023. DOI : https://www.doi.org/10.56726/IRJMETS35340

a. Lakshmi Rishika, Ch. Aishwarya, A. Sahithi, M. Premchender. Real-time Vehicle Detection and Tracking using YOLO-based Deep Sort Model: A Computer Vision Application for Traffic Surveillance. Turkish Journal of Computer and Mathematics Education Vol.14 No.01 (2023),255- 264. DOI: https://doi.org/10.17762/turcomat.v14i1.13530

[10] Hao Yi, Bo Liu, Bin Zhao, Enhai Liu. Small Object Detection Algorithm Based on Improved YOLOv8 for Remote Sensing. January 2023 IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing PP(99):1-15 DOI: https://doi.org/10.1109/JSTARS.2023.3339235

[11] D.T. Mane, S. Sangve, S. Kandhare, Real-Time Vehicle Accident Recognition from Traffic Video Surveillance using YOLOV8 and OpenCV. International Journal on Recent and Innovation Trends in Computing and Communication DOI: https://doi.org/10.17762/ijritcc.v11i5s.6651

[12] D. T. Mane, P. Kumbharkar, N. Earan, K. Patil, S. Bonde. A Research Survey on Real-Time Intelligent Traffic System. International Journal of Emerging Technology and Advanced Engineering. DOI: https://doi.org/10.46338/ijetae0423_0

[13] Roboflow Blog

[14] R. Wang, N. Sang, R. Huang, et al., License plate detection using gradient information and cascade detectors, Optik-International Journal for Light and Electron Optics , 186–190.

[15] J. Jiao, Q. Ye, and Q. Huang, A configurable method for multi-style license plate recognition, Pattern Recognition , 358–369.

[16] H. Li and C. Shen, Reading car license plates using deep convolutional neural networks and lstms, arXiv preprint arXiv:1601.05610.

[17] H. Li et. al. License Plate Detection Using Convolutional Neural Network. 2017 3rd IEEE International Conference on Computer and Communications

[18] N. Palanviel AP. et.al. Automatic Number Plate Detection in Vehicles using Faster R-CNN. IEEE ICSCAN