Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 9 No. 3 (2025) 544 - 553 e-ISSN: 2580-0760

Automated Indonesian Plate Recognition: YOLOv8 Detection and TensorFlow-CNN Character Classification

Windu Gata[1*], Dwiza Riana[2], Muhammad Haris[3], Maria Irmina Prasetiyowati[4], Dika Putri Metalica[5]
[1,2,3,5]Computer Science, Faculty of Information Technology, Universitas Nusa Mandiri, Jakarta, Indonesia
[4]Informatics, Faculty of Engineering & Informatics, Universitas Multimedia Nusantara, Jakarta, Indonesia

[1]windu@nusamandiri.ac.id, [2]dwiza@nusamandiri.ac.id, [3]muhammad.uhs@nusamandiri.ac.id, [4]maria@umn.ac.id, [5]dika.putri93@gmail.com

Abstract

The precise identification and reading of Indonesian vehicle number plates are important in many areas, including the enforcement of law, collection of charges, management of parking areas, and safety measures. This study integrates the implementation of the YOLOv8 object detection algorithm with three OCR methods: EasyOCR, TesseractOCR, and TensorFlow. YOLOv8 is capable of identifying license plates from images and videos at a high speed and reliability under different conditions and therefore is used in this study to perform plate detection in images and videos. After licenses are detected, OCR techniques are performed to segment and read the letters. Both EasyOCR and TesseractOCR performed moderately well on static images achieving accuracy rates of 70% and 68% respectively, but both suffered significantly lower performance in video scenarios. Of the 100 video frames, EasyOCR was able to correctly identify characters in 61 frames and TesseractOCR in 58 frames, while the TensorFlow-based model outperformed the other two with 75 correct recognitions. Furthermore, easy OCR and static images as input while the TensorFlow-based models completed them with 100% accuracy. This observation can be explained by its design, which utilizes a CNN with ReLU activation and Softmax outputs, trained on 10,261 annotated characters and was enhanced with five different data augmentation techniques. The model shows strong performance in its ability to handle dynamic conditions such as motion blur, changing light conditions, and rotation of the plate angle. The results underscore the drawbacks of one-size-fits-all OCR applications in real-world use cases and stress the need for bespoke model training, as well as hierarchical contouring, in the context of automatic license plate recognition (ALPR). This study provides additional insights into ALPR systems by delivering a robust, scalable, and real-time tool for plate and character recognition, which is essential for intelligent transportation systems.

Keywords: YOLO; TensorFlow; optical character recognition (OCR); indonesian license plate detection; deep learning

How to Cite: Wndu Gata, Dwiza Riana, Muhammad Haris, Maria Irmina Prasetiyowati, and Dika Putri Metalica, "Automated Indonesian Plate Recognition: YOLOv8 Detection and TensorFlow-CNN Character Classification", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 9, no. 3, pp. 544 - 553, Jun. 2025.
Permalink/DOI: https://doi.org/10.29207/resti.v9i3.6310

Received: January 16, 2025
Accepted: June 4, 2025
Available Online: June 15, 2025

1. Introduction

Effective recognition of Indonesian vehicle registration plates is essential for the enforcement of regulations and for ensuring public safety. The rigorous monitoring of violation offenses such as over-speeding, illegal parking, or driving into restricted areas presents growing difficulties, which can be addressed with the ALPR (Automatic License Plate Recognition) system due to its scalable capabilities. These systems enhance automated detection of offenders, minimizing the need for manual processes and increasing efficiency in enforcement operations. Furthermore, the detection of vehicle registration plates also serves the purpose of toll fee collection, parking management, and security surveillance where immediate identification of vehicles is required.

The classical methods of image processing continue to be useful in ALPR systems because of their straightforwardness and flexibility. One research employed morphological techniques followed by KNN to solve recognition issues for Indian plates due to their unconventional shapes [1]. The combination of Gaussian noise removal along with contour-based segmentation contributed towards reliable detection of plates in vehicles [2]. Another work applied contouring and edge detection to address problems with

illumination differences [3]. Pipelines designed for plate recognition and traffic management have been implemented using OpenCV, proving adaptability for real-life traffic situations [4]. Morphological methods have been applied to achieve high accuracy for Indonesia's plate recognition [5]. Stroke width transform and neural networks were applied to number plate detection and classification. Morphological preprocessing helped in the recognition of candidate plates [6]. Plate recognition together with character segmentation was enhanced using a combination of histogram equalization and Sobel edge detection [7].

The You Only Look Once (YOLO) algorithm for object detection has found application in License Plate Recognition (LPR) systems due to its efficiency and precision. In a comparison study on the versions of YOLO (v5, v7, v8, v9), it was reported that YOLOv8 outperformed all others in precision and recall, especially under varying lighting conditions [8]. Other researches also show that the integration of YOLO with EasyOCR achieved 94% accuracy for parking system plate recognition [9]. Moreover, YOLOv8 was also incorporated into the SAMBARA system aimed at automating the tax verification and counterfeit plate detection processes in Indonesia [10]. As for other applications of YOLO, one study combined YOLO with ResNet to improve parking management systems [11]. YOLOv5 also found use in traffic monitoring and parking systems with remarkable accuracy in detection [12]. A combination of EasyOCR, Tesseract OCR, and YOLOv5 enabled a recognition accuracy of 95% [13]. YOLOv8 customized for Iranian motorcycle plates achieved 99% detection accuracy [14]. Further enhancements to YOLOv5 with GRU for character recognition increased the accuracy to 98.98% [15]. Advanced versions of YOLOv5 developed for Indian plates detected them with 99.1% accuracy [16]. Another system incorporated YOLOv5 into Secure Park, a real-time intrusion detection system designed with the Microsoft Vision API and OCR [17]. Real-time detection optimization for urban traffic environments were addressed, including background noise, and obstacles covering the plates were tackled [18]. Extensive work has been done towards Indonesian LPR systems, one example being a 94% accurate detection using a YOLOv8 and EasyOCR-based system, proving resilience in identifying non-conventional plates under difficult circumstances [19].

The ease of performing license plate recognition (LPR) operations has significantly improved with the introduction of convolutional neural networks (CNNs), primarily because these systems are now capable of overcoming substantial environmental noise. Robust Recognition OCR technology incorporated CNNs to enhance recognition accuracy during variable conditions [20]. Recognition of blurred license plates within Indonesia was enhanced through the use of multi-scale CNN models [21]. Character recognition for noisy data was improved with the use of sliding windows and CNNs [22]. The ability of CNNs to handle noise was illustrated through his robust recognition of Indonesian plates with obstructions [23]. Neural processing has shown versatility to different plate shapes and angles which would allow for diverse conditions of implementation [24]. Further refinements to segmentation techniques CNNs provided enhanced precision in noisy environments[22].

ALPR systems are getting more traffic fines and infractions enforcement violations using ALPR Automated license plate recognition systems integrated with traffic safety systems. One implementation that issued graduated penalties for road behavior OCR based recognition of helmet riding showed restriction improved road safety. License and insurance verification enforcement automation has also been proposed as non-compliance augmented issuing penalties OCR zero regard pay [25]. The systems developed in these studies have proven the effectiveness of LPR in advancing the goal of automating the enforcement of traffic laws. Comparison studies have been very productive in understanding the diverse advantages and disadvantages of different LPR systems. Work on Indian plates is documented in the form of review focusing on traditional and YOLO approaches [26]. There are also comparisons of YOLO and traditional edge detection that demonstrate the importance of deep learning to surveillance systems [27].

Existing works show that OCRs like EasyOCR and TesseractOCR make use of YOLO framework for license plate detection. Thus, this study has two objectives: first, to implement the YOLO algorithm for vehicle license plate detection in images and videos, and second, to evaluate OCRs EasyOCR, TesserOCR and those implemented with TensorFlow. Moreover, this study analyzes the dynamic video footage of the license plates to test algorithms for character recognition in real-time scenarios. For each image of the license plate, three main parts are defined: the regional code (letters) as the first segment, the plate number (digits) as the second segment, and sub-regional code (letters) as the third segment. This research intends to devise methods of recognizing license plate characters, particularly from video data, so as to analyze the effectiveness of OCRs and the segmentation comparison with the models from TensorFlow.

Most of the existing YOLO-OCR implementations target the US, Europe and India which have different standards from Indonesia and are thus, poorly adapted to the Indonesian format. These models are not equipped to deal with the alphanumeric strings, fonts, spacing, and region code variations that are unique to Indonesian plates.

This study focuses on these limitations by utilizing a segmented method based on Indonesia's tectonic plate configuration, in conjunction with a TensorFlow model.

2. Methods

This study involved several steps in relation to the objectives of license plate recognition and character recognition. The first step uses the YOLO algorithm to detect license plates because of its speed and accuracy in locating plates within images or video frames. After detection, the next step is to segment the detected plate into three parts: the regional code of the plate's letters, the numeric portion of the plate as digits, and the sub-regional code which are also letters.

Character recognition involves comparative analysis using three approaches: EasyOCR, TesseractOCR and one based on TensorFlow. These approaches will be tested for their ability to recognize characters in still and video images. To assess the robustness of the system, video analysis is conducted in real time, simulating actual working conditions that test the algorithms under varying light, angle, and environmental changes. Each OCR method's results are benchmarked against one another on accuracy, processing time, and dependability, with the ultimate goal being the identification of the most suitable method for the recognition of dynamically changing conditions. This is illustrated in Figure 1.

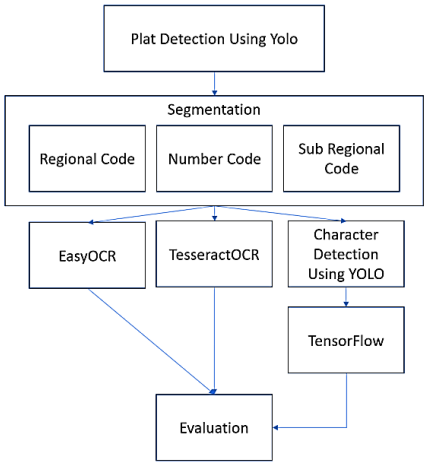

Figure 1. Comparative Model Plat Number Research Process

2.1 License Plate Detection with YOLOv8

The specifications of the device detail its parts and components, highlighting a 12th Gen Intel(R) Core (TM) i7-12700 2.10 GHz processor, 32.0 GB of RAM with 31.8 GB usable, and an NVIDIA GeForce RTX 3060 Ti GPU. The sistem is configured with Windows 11 Operating System and uses Python version 3.9.2 alongside YOLOv8 for object detection, Tesseract 0.3.13 for OCR, and EasyOCR 1.7.1 for advanced text recognition.

The initial dataset consists of 861 images, with each image annotated with bounding boxes to mark license plates. This annotation process is conducted using tools like YOLO and LabelIMG to ensure accurate localization of license plates. The model handler uses YOLO (You Only Look Once) to either initialize a new model or load an existing one. If the specified model file ends with `.yaml`, it initializes a new YOLO model; if it ends with `.pt`, it loads a pre-trained model.

This optimization process is expressed as shown in Equation 1:

$$\theta opt = arg\ min\ \theta\ \sum_{i=1}^{E} L(M_\theta(x_i), y_i) \quad (1)$$

Where x_i represents the input images, y_i represents the corresponding labels, M_θ is the model parameterized by θ, and L is the loss function. This equation describes the process of finding the optimal model parameters θopt by minimizing the cumulative loss over a dataset of $\sum$ examples.

The model evaluation on a validation dataset V is given by Equation 2.

$$P = M_{pretrained}\ (V) \quad (2)$$

where P represents the performance metrics, and $M_{pretrained}$ is the pre-trained model applied to the validation dataset V. This step computes metrics to assess the model's accuracy and effectiveness.

The prediction process for new input images I is expressed as Equation 3.

$$\hat{y} = M_{pretrained}\ (I) \quad (3)$$

where $\hat{y}$ represents the bounding box predictions for license plate localization, and $M_{pretrained}$ is the pre-trained model applied to the input images I.

2.2 Segmentation

For a clearer understanding of the segmentation of Indonesian license plates, refer to Figure 2. In this illustration, Segment 1 represents the regional code, such as the letter "B," Segment 2 represents the numeric identifier, such as "6401," and Segment 3 represents the sub-regional code, such as "VRL". This research focuses solely on license plates and does not extend to tax-related aspects, as indicated by the numbers "08.2028" displayed on the plate.

Figure 2. Sample Segmentation Indonesian Plate Number

The process involves license plates successfully detected by YOLO with a confidence percentage of 50% being standardized to a pixel size of 400 x 200 and divided into three segments. Segment 1 covers the pixel range [10:128, 10:60], Segment 2 covers [10:128, 60:238], and Segment 3 covers [10:128, 238:400].

2.3 OCR

OCR is a technology that converts physical text into digital data. In addition to its use in finance, healthcare, and education, OCR supports traffic management by

reading vehicle license plates and detecting violations. This technology enhances efficiency and accessibility, and drives digital transformation across various fields.

EasyOCR and Tesseract were selected due to their widespread adoption and open-source availability. TensorFlow was chosen to explore a custom-trained deep learning approach. Other advanced frameworks such as PaddleOCR or Vision Transformers were not included due to complexity in model customization and training requirements outside the study's scope.

2.3.1 EasyOCR

EasyOCR is a powerful OCR tool developed by Jaided AI. Supporting over 80 languages, it's built with PyTorch for efficient text extraction from images, scanned documents, or handwriting. EasyOCR is ideal for tasks like reading license plates, processing invoices, and automating text extraction across industries.

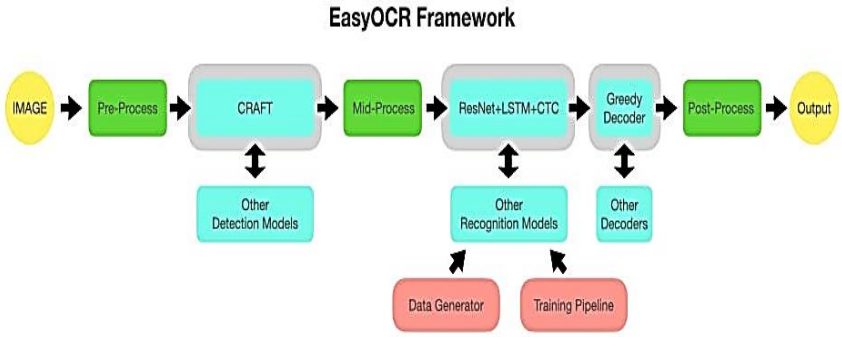

Figure 3. EasyOCR Framework

The EasyOCR framework Figure 3 is composed of several important processes for Optical Character Recognition. It starts with pre-process, which enables the corresponding image to be input for the text to be detected. The CRAFT Character Region Awareness For Text detection model recognizes the regions that contain text, then moves to Mid-Process stage to refine these regions. Text recognition is done using the Residual Network (ResNet), Long Short-Term Memory (LSTM), and Connectionist Temporal Classification (CTC) pipeline along with other recognition models. Patterns are converted into text and the decoding is done using greedy decoders or other decoders which translate recognized patterns into text.

Events defined above are all finalizing processes which can be done after text is recognized and a final output is required. These outputs can be refined further which leads to better uses of OCR systems. The accuracy of machine-read text is enhanced with the addition of a Data Generator and a Training Pipeline under the EasyOCR framework.

2.3.2 TesseractOCR

Tesseract OCR is an Optical Character Recognition system, which is one of the systems available free of charge on the internet. It was first offered by the firm Hewlett-Packard and was later enhanced by Google. The program captures hand-written documents or printed texts found in images and documents transforming them into texts which can be edited electronically. Tesseract offers flexibility, conceptualized document scanning, automated text processing and customization which in return benefits a wide range of use.

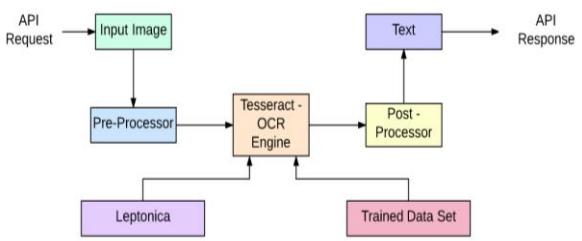

Figure 4. The process of Tesseract OCR

In Figure 4, the Tesseract OCR process begins with an input image sent via API request. The image undergoes preprocessing using Leptonica for quality enhancement. The Tesseract engine, supported by trained data sets, extracts text from the processed image. Post-processing then refines the output, delivering accurate text as the final API response.

2.3.3 Character Detection using YOLO and TensorFlow

Character Detection on License Plates: The next phase involved identifying the individual characters on the license plate. At this stage, a secondary instance of YOLO was employed, specifically for isolating characters within the plate's boundaries. Each alphanumeric character, whether a letter or a digit, was enclosed in its own bounding box, ensuring that each character was distinctly separated from the others.

Character Recognition with TensorFlow: In the final stage, the system focused on recognizing each of the detected characters and classifying them as the correct letter or number, see Figure 5.

In this research, a dataset of 10,261 annotated characters from Indonesian license plates was manually collected

from images and videos. The data was taken from various regions in Indonesia to ensure diversity in plate formats, you can see the datasets on this dataset. Labeling was performed manually using LabelImg, with separate bounding boxes for each character.

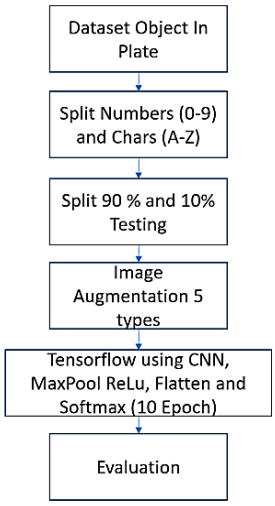

Figure 5. The Process of TensorFlow

This process could be summarized through a single formula-like expression that outlined the series of transformations the input image underwent as it passed through the layers of the CNN.

The CNN model started with an input layer that represented the image with a height and width of 64 pixels and 3 colour channels (RGB).

This could be expressed mathematically as shown in Equation 4:

$$X \in R^{64} \times^{64} \times^{3} \quad (4)$$

This convolution produced an output with dimensions of 62×62 and 32 channels, as the image size slightly decreased due to the application of filters without padding. Subsequently, the bias b1 was added to the convolution result, and the ReLU activation function was applied. The ReLU function transformed all negative values to zero, introducing non-linearity to the output, referred to as H1. The final output H1 had dimensions of 62×62×32, where 32 represented the number of output channels, capturing basic features detected from the input image. This process was expressed in Equation 5:

$$H_1 = \text{ReLU}(X * W_1 + b_1), H_1 \in R^{62} \times^{62} \times^{32} \quad (5)$$

The first max-pooling layer applied 2×2 pooling, reducing the spatial dimensions of the output while retaining the 32 channels from the previous layer. This operation was expressed in Equation 6:

$$H_2 = \text{MaxPool}(H_1), H_2 \in R^{64} \times^{64} \times^{32} \quad (6)$$

The second convolutional layer applied 64 filters of size 3×3 with ReLU activation, generating feature maps. This was followed by a second max-pooling layer that used 2×2 pooling, further reducing the spatial dimensions. The operations for this layer were expressed mathematically in Equation 7:

$$H_3 = \text{ReLU}(H_2 * W_2 + b_2), H_3 \in R^{29} \times^{29} \times^{64} \quad (7)$$

This resulted in an output with dimensions of 14×14×64, where 14×14 was the reduced spatial size, and 64 was the number of channels, consistent with H3. This operation was expressed as shown in Equation 8:

$$H_4 = \text{MaxPool}(H_3), H_4 \in R^{14} \times^{14} \times^{64}) \quad (8)$$

In the third convolutional layer, the output tensor had dimensions of 12×12×128, where 12×12 represented the spatial dimensions after applying the convolution, and 128 was the number of output channels corresponding to the filters. This operation was expressed in Equation 9:

$$H^5 = ReLU(H^4 * W^3 + b^3), where\ H^5 \in R^{12} \times^{12} \times^{128} \quad (9)$$

In the next step, the output dimensions were reduced to 6×6×128. The spatial dimensions (6×6) resulted from the pooling operation, which shrank the width and height of H5 from 12×12 to 6×6. The depth (128 channels) remained unchanged, preserving the feature complexity learned in the previous layer. This operation was expressed in Equation 10:

$$H_6 = \text{MaxPool}(H_5), \text{where } H_6 \in R^{6} \times^{6} \times^{128} \quad (10)$$

Convert the 3D tensor H_6 into a 1D vector with a size of 4608, preparing it for input into the fully connected layers. This transformation is expressed in Equation 11:

$$f = \text{Flatten}(H_6), \text{where } f \in R^{4608} \quad (11)$$

W4: The weight matrix for this dense layer, consisting of weights that the model learned to connect each input feature in f to each of the 128 units. b4: The bias term for the dense layer, which adjusted the output before the activation function was applied. ·: Represented matrix multiplication between f and W4. The operation of this dense layer was described by Equation 12:

$$h_7 = \text{ReLU}(f \cdot W_4 + b_4), h_7 \in R^{128} \quad (12)$$

Finally, the output layer consisted of a dense layer with 10 units for numbers and 36 units for characters, represented as nnn. A softmax activation function was applied to produce probabilities for each class. This operation was described in Equation 13:

$$y = \text{Softmax}(h_7 \cdot W_5 + b_5), y \in R^{n} \quad (13)$$

The test set was held out and never used during training or validation, ensuring a fair evaluation. This division was applied to address specific challenges that arose when identifying and classifying characters, especially when dealing with problematic or ambiguous symbols such as (~).

The ImageDataGenerator was employed to enhance the training data by augmenting images through transformations that helped the model generalize better.

By rotating, shifting, and zooming the images, the generator simulated real-world variations that were particularly beneficial for dealing with the diversity of characters and numbers in different situations. The robustness of the dataset was achieved through transformations, such as horizontal and vertical shifts and slight image shearing. Gaps arising from these transformations were filled using nearest-neighbor filling, which maintained uniformity, allowing for the effective augmentation of image information without loss. This configuration enhances model dependability, particularly in the context of unseen data, improving the accuracy in diverse character recognition scenarios.

Table 1. Training and Validation Results of CNN + Softmax over 10 Epochs for Character Recognition

Epoch	Loss	Accuracy	Val_Loss	Val_Accuracy
1	0.2548	0.9279	0.1887	0.9641
2	0.0581	0.9838	0.2252	0.9575
3	0.0345	0.9906	0.2768	0.9608
4	0.0244	0.9933	0.3044	0.9700
5	0.0220	0.9944	0.3559	0.9711
6	0.0160	0.9959	0.2714	0.9758
7	0.0160	0.9960	0.3324	0.9707
8	0.0129	0.9967	0.2765	0.9729
9	0.0105	0.9972	0.2940	0.9667
10	0.0116	0.9973	0.3625	0.9676

Table 1 represented the CNN model in this study which was trained over 10 epochs using a license plate character dataset, which was divided into digits (0–9) and letters (A–Z). The dataset was split into 90% for training and 10% for testing. To enhance model generalization, five types of image augmentation techniques were applied to the training set.

3. Results and Discussions

3.1 License Plate Detection with YOLOv8

As shown in Figure 6, based on the provided F1-Confidence Curve and detection results, the model demonstrated high performance in license plate recognition. The F1-Confidence Curve indicated that the model achieved a peak F1 score close to 1.0, with the F1 score for all classes reaching approximately 0.99 at a confidence threshold of 0.547.

This result reflected a strong balance between precision and recall, showing the model's ability to accurately detect license plates under various conditions.

Figure 7 displayed multiple images of detected license plates, each highlighted with a blue bounding box labeled "license," indicating successful detection. The model was able to effectively identify plates across various angles, lighting conditions, and vehicle types, demonstrating its robustness in real-world settings.

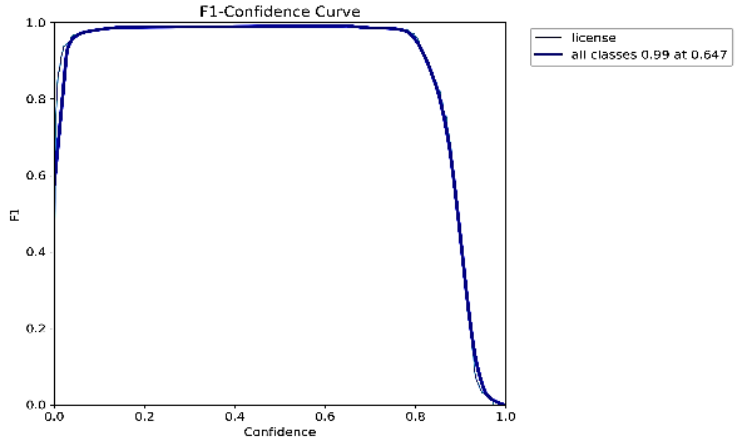

Figure 6. F1- Score - Identification Plate from Figure

Figure 7. Sample Plate from Frame

3.2 EasyOCR

EasyOCR was utilized to recognize each character for sections 1, 2, and 3. With static or non-moving data, EasyOCR accurately recognized 602 out of 861 license plates, which was a success rate of 70%. However, for dynamic data, like video footage, EasyOCR had difficulty with accuracy in character-by-character recognition.

This issue was caused by movement within the images resulting in blurriness and lack of sharpness, which severely hindered recognition accuracy. This demonstrated that the effectiveness of OCR, especially with background motion or fluctuating light conditions, heavily relied on the quality of input data. The result could be observed in this video.

3.3 TesseractOCR

As with segments 1, 2, and 3, Tesseract OCR was used for character recognition. For static data, Tesseract OCR recognized 585 license plates out of 861 with an accuracy of 68%. However, when tested with dynamic

data (video data), performance dropped significantly, just like with EasyOCR, due to blurriness and loss of detail from motion smoothing.

This showed that both Tesseract OCR and EasyOCR were greatly influenced by the quality of data provided, particularly in situations where there was movement or visual instability, which adversely impacted precision and effectiveness. The results could be viewed in this video.

3.4 Character Detection within Plates using YOLO

From character-level detection results on plates, it could be noticed from the F1-Confidence Curve (Figure 8) that the model was able to detect all characters on the plate with a total of 10,261 characters. Furthermore, the curve demonstrated an F1 score of a rather high 0.97 for all classes at a 0.527 confidence threshold. This showed that at this threshold, the model captured an excellent blend of precision and recall, which resulted in high accuracy of character detection. The F1 score remained high with increasing confidence levels, up until a sharp drop, suggesting that while the model accurately detected characters within certain parameters, it struggled beyond those parameters. Altogether, the graph indicated model robustness in terms of precision and recall when detecting specific characters for license plates.

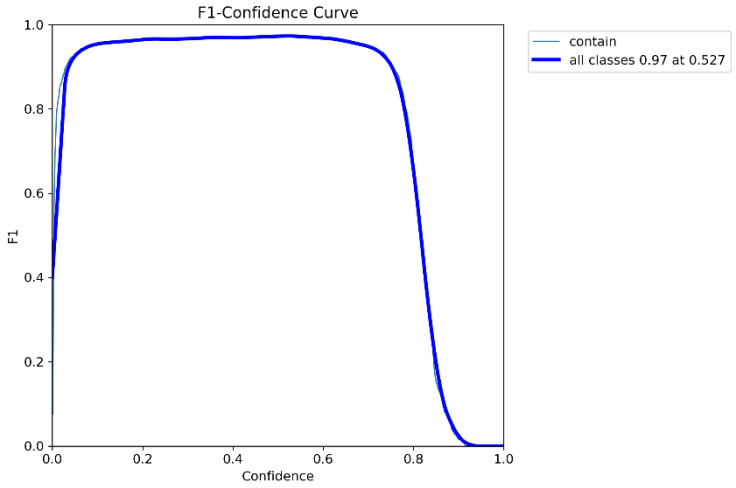

Figure 8. F1 Score for Detection Characters on the Plate

Figure 9. Sample Characters Detection on the Plate

Sample images of characters detected on license plates were shown in Figure 9. All characters embedded within the license plates were enclosed within bounding boxes marked as "contain," signaling that each alphanumeric element had been precisely detected and successfully segmented. The various license plates captured at different angles and under different light conditions were reliably processed with all characters identified and separated.

This showcased the model's strong capabilities in detailed character-level recognition, which was important for accurate recognition of license plates. The uniform bounding illustrated that the model could differentiate discrete characters, permitting meta-analysis character-by-character, which was critical for vehicle identification and automated plate recognition in real-life applications.

TensorFlow assisted in detecting entire static license plates in a given image. In dynamic scenarios, this algorithm was capable of number detection with YOLO performing character detection, and it worked remarkably well even under difficult conditions.

As shown in Figure 10, one of the frames from the implementation video demonstrated how the model tracked and recognized license plates on motorcycles within a moving stream of traffic. License plate detection occurred on two motorcycles as bounding boxes encapsulating the numbers were drawn and every character was labeled in green. This frame showcased the capabilities of the model to track and recognize letters in a complex background containing a lot of motion, dynamic perspectives, and elements of nature. The precision and segmentation in such a complex environment demonstrated the model's versatility and accuracy for real-time traffic video analysis.

Figure 10. Frames of Implementation Video on the Street

In Figure 11, several samples of detected license plates are shown, where each plate is segmented and each character is enclosed within a green bounding box. The model successfully detects and isolates individual characters on various plates, displaying a clear and consistent identification of letters and numbers.

This sample set demonstrated the model's capability to handle different plate formats and orientations, accurately recognizing characters despite variations in font, spacing, and slight image distortions. The precision of the bounding boxes around each character indicated the model's robustness in distinguishing characters, which was essential for applications requiring detailed analysis, such as automated license plate recognition in traffic systems. For detailed analysis on this video.

Figure 11. Sample Detect Plate Number

3.5 Evaluation

The comparative study on EasyOCR, TesseractOCR, and TensorFlow in conjunction with YOLO for license plate recognition offered important observations regarding OCR efficacy in different contexts. While the three approaches had identical F1 scores of 0.97 for the YOLO-based facility of license plate detection, character recognition performed quite differently. This underscored that OCR systems should not solely be evaluated on detection performance but also on the fidelity of recognition in both static and dynamic contexts.

Table 2. Evaluation Using Static Figure and Video

OCR Method	Algorithm	Dataset Character & Accuracy	Image	Video Per 100 Frames
EasyOCR	ResNet, LSTM, CTC	-	70%	61%
TesseractOCR	Tesseract Engine (Hewlett Packard + Google)	-	68%	58%
TensorFlow	CNN + Relu + SoftMax	10.261 10 Epoch Accuracy 99.7%	100%	75%

In Table 2, a comparison was made between three OCR systems with respect to character recognition in license plates from still and animated images (videos). In the case of still images, EasyOCR scored 70%. TesseractOCR scored slightly lower at 68%. TensorFlow scored 100%, supported by a training dataset for 10 epochs which gave a training accuracy of 99.7%. In video data (based on 100 test frames), EasyOCR recognized characters in 61 frames, TesseractOCR in 58 frames, and TensorFlow in 75 frames.

These results demonstrated that while EasyOCR and TesseractOCR performed well with static images, their performance dropped substantially in dynamic video—likely due to motion blur and changes in lighting and frame stability. On the other hand, TensorFlow demonstrated stronger robustness and generalization, maintaining high accuracy even in real-world video environments. This made TensorFlow the most effective OCR method among the three for implementing automated license plate recognition systems, particularly in dynamic, real-time footage.

Failure cases were observed primarily in video frames with heavy motion blur or partial occlusion. EasyOCR and Tesseract frequently misclassified characters with similar shape. YOLO occasionally missed plates at extreme angles. These findings indicated the need for enhanced pre-processing and real-time frame stabilization in future deployments.

While TensorFlow demonstrated the highest accuracy, a statistical t-test was not conducted to verify the significance of performance differences. This will be explored in future work to confirm reliability across various scenarios.

3.6 Processing Time Comparison

The average time required to process a single static image was approximately 0.14 seconds for EasyOCR, 0.22 seconds for Tesseract, and 0.27 seconds for TensorFlow. For video data (per frame), TensorFlow required the longest due to sequential character segmentation, averaging 0.35 seconds per frame.

4. Conclusions

This study investigated the detection and recognition of Indonesian license plates by integrating YOLO for object detection and three OCR methods: EasyOCR, TesseractOCR, and TensorFlow for character recognition. YOLOv8 demonstrated strong detection capabilities across both static images and video data, effectively localized license plates under real-world conditions.

In the character recognition phase, each OCR method showed varying levels of performance. EasyOCR and TesseractOCR achieved moderate accuracies of 70 percent and 68 percent on static images. However, their performance significantly dropped when applied to dynamic video frames, with accuracy decreased to 61 percent and 58 percent out of 100 frames respectively. On the other hand, the TensorFlow-based CNN model achieved 100 percent accuracy on static images and 75 percent accuracy on dynamic video.

These results indicated that the TensorFlow model, supported by CNN architecture with ReLU and Softmax layers, was more reliable and adaptable than traditional OCR engines. The study also highlighted the importance of structured plate segmentation and a well-prepared dataset in improving OCR accuracy.

Beyond selecting the best-performing OCR, future work would explore hybrid approaches combining rule-based filtering with deep learning, adaptive segmentation for plates with variable layouts, and the integration of transformer-based OCR models for enhanced contextual understanding and this system could be extended to support multi-object tracking, behaviour analysis, and improved performance in challenging conditions such as low light, occlusions, or adverse weather.

While the TensorFlow model achieved exceptional performance on static images (100% accuracy), the diverging trend between training and validation loss during model training, along with the significant performance drop in video scenarios (75%), suggested potential mild overfitting. The 3% gap between final training accuracy (99.73%) and validation accuracy (96.76%) indicated the model might have partially memorized training patterns. However, the reasonable video performance suggested the model had learned generalizable features. The overfitting appeared manageable but needed to be addressed in future work through better regularization and validation strategies. Future implementations should have incorporated early stopping mechanisms, cross-validation techniques, and enhanced regularization methods to improve model generalization and ensure more robust performance across diverse real-world conditions.

References

[1] P. R. K. Varma, S. Ganta, B. K. H., and P. Svsrk, "A novel method for Indian vehicle registration number plate detection and recognition using image processing techniques," *Procedia Computer Science*, vol. 167, pp. 2623–2633, 2020, doi: 10.1016/j.procs.2020.03.324.

[2] K. Anusha, S. Nachiyappan, M. Braveen, K. V. Pradeep, and S. R. Yarlagadda, "A simple number plate detection technique with support vector machine for on-road vehicles," in *2022 International Virtual Conference on Power Engineering Computing and Control (PECCON)*, Chennai, India, May 2022, pp. 1–6. doi: 10.1109/PECCON55017.2022.9851000.

[3] R. Boliwala and M. Pawar, "Automatic number plate detection for varying illumination conditions," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, Melmaruvathur, India, Apr. 2016, pp. 658–661. doi: 10.1109/ICCSP.2016.7754224.

[4] R. R. Chandrika and others, "Number plate recognition using OpenCV," in *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*, Pune, India, Mar. 2024, pp. 1–4. doi: 10.1109/ESCI59607.2024.10497253.

[5] Y. P. Pasrun, M. Muchtar, A. N. Basyarah, and Noorhasanah, "Indonesian license plate detection using morphological operation," *IOP Conference Series: Materials Science and Engineering*, vol. 797, no. 1, p. 012037, Mar. 2020, doi: 10.1088/1757-899X/797/1/012037.

[6] I. M. Gorovyi and I. O. Smirnov, "Robust number plate detector based on stroke width transform and neural network," in *2015 Signal Processing Symposium (SPSympo)*, Debe, Poland, Jun. 2015, pp. 1–4. doi: 10.1109/SPS.2015.7168289.

[7] G. Kothai, E. Povammal, A. S., and D. V., "An efficient deep learning approach for automatic license plate detection with novel feature extraction," *Procedia Computer Science*, vol. 235, pp. 2822–2832, 2024, doi: 10.1016/j.procs.2024.04.267.

[8] A. C. Bukola, P. A. Owolawi, C. Du, and E. Van Wyk, "A systematic review and comparative analysis approach to boom gate access using plate number recognition," *Computers*, vol. 13, no. 11, p. 286, Nov. 2024, doi: 10.3390/computers13110286.

[9] N. Supriya and others, "An efficient license plate recognition model through deep learning integration with YOLO and OCR techniques," in *2024 International Conference on Advances in Modern Age Technologies for Health and Engineering Sciences (AMATHE)*, Shivamogga, India, May 2024, pp. 1–5. doi: 10.1109/AMATHE61652.2024.10582121.

[10] A. U. Nawawi and others, "Automatic license plate recognition: Automated tax verification for registered vehicles via SAMBARA server," in *2024 10th International Conference on Wireless and Telematics (ICWT)*, Batam, Indonesia, Jul. 2024, pp. 1–5. doi: 10.1109/ICWT62080.2024.10674718.

[11] J. Joshua, J. Hendryli, and D. E. Herwindiati, "Automatic license plate recognition for parking system using convolutional neural networks," in *2020 International Conference on Information Management and Technology (ICIMTech)*, Bandung, Indonesia, Aug. 2020, pp. 71–74. doi: 10.1109/ICIMTech50083.2020.9211173.

[12] I. Varalakshmi, M. K. Santhoshi, and S. Swetha, "Automatic number plate recognition system using deep learning YOLOv5 algorithm," in *2023 International Conference on System, Computation, Automation and Networking (ICSCAN)*, Puducherry, India, Nov. 2023, pp. 1–6. doi: 10.1109/ICSCAN58655.2023.10395604.

[13] D. R. Vedhaviyassh and others, "Comparative analysis of EasyOCR and TesseractOCR for automatic license plate recognition using deep learning algorithm," in *2022 6th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, Dec.

2022, pp. 966–971. doi: 10.1109/ICECA55336.2022.10009215.

[14] A. Fathi, B. Moradi, I. Zarei, and A. Shirbandi, "Deep learning-based system for automatic motorcycle license plates detection and recognition," *Signal, Image and Video Processing*, vol. 18, no. 12, pp. 8869–8879, Dec. 2024, doi: 10.1007/s11760-024-03514-5.

[15] H. Shi and D. Zhao, "License plate recognition system based on improved YOLOv5 and GRU," *IEEE Access*, vol. 11, pp. 10429–10439, 2023, doi: 10.1109/ACCESS.2023.3240439.

[16] A. Kathirvel and others, "Systematic number plate detection using improved YOLOv5 detector," in *2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*, Vellore, India, May 2023, pp. 1–6. doi: 10.1109/ViTECoN58111.2023.10157727.

[17] D. U. Nayak and others, "SecurePark: Vehicle intrusion detection system," in *2021 Asian Conference on Innovation in Technology (ASIANCON)*, Pune, India, Aug. 2021, pp. 1–6. doi: 10.1109/ASIANCON51346.2021.9544923.

[18] X. Li, "Real-time license plate number detection based on image contour," *Journal of Physics: Conference Series*, vol. 1650, no. 3, p. 032073, Oct. 2020, doi: 10.1088/1742-6596/1650/3/032073.

[19] A. D. Iriawan and A. Sunyoto, "Automatic license plate recognition system in Indonesia using YOLOv8 and EasyOCR algorithm," in *2023 6th International Conference on Information and Communications Technology (ICOIACT)*, Yogyakarta, Indonesia, Nov. 2023, pp. 384–388. doi: 10.1109/ICOIACT59844.2023.10455908.

[20] S. Khan and others, "A computer vision-based vehicle detection system leveraging deep learning," in *2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC)*, Tandojam, Pakistan, Jan. 2024, pp. 1–7. doi: 10.1109/KHI-HTC60760.2024.10482163.

[21] U. L. Yuhana, G. Edo, and H. Syarif, "Enhancement of blurred Indonesian license plate number identification using multi-scale information CNN," in *2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, Bangalore, India, Dec. 2023, pp. 1–6. doi: 10.1109/SMARTGENCON60755.2023.10442912.

[22] A. T. Musaddid, A. Bejo, and R. Hidayat, "Improvement of character segmentation for Indonesian license plate recognition algorithm using CNN," in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia, Dec. 2019, pp. 279–283. doi: 10.1109/ISRITI48646.2019.9034614.

[23] I. W. Notonogoro, Jondri, and A. Arifianto, "Indonesian license plate recognition using convolutional neural network," in *2018 6th International Conference on Information and Communication Technology (ICoICT)*, Bandung, Indonesia, May 2018, pp. 366–369. doi: 10.1109/ICoICT.2018.8528761.

[24] S. J. Fusic, S. Karthikeyan, H. Ramesh, and A. Subbiah, "Vehicle license plate detection and recognition using neural network," in *2020 4th International Conference on Computing, Communication and Signal Processing (ICCCSP)*, Chennai, India, Sep. 2020, pp. 1–5. doi: 10.1109/ICCCSP49186.2020.9315206.

[25] A. Rocque and others, "Enhancing traffic violation enforcement: A system utilizing helmet and number plate detection," in *2023 OITS International Conference on Information Technology (OCIT)*, Raipur, India, Dec. 2023, pp. 598–604. doi: 10.1109/OCIT59427.2023.10431231.

[26] C. K. Sahu and others, "A comparative analysis of deep learning approach for automatic number plate recognition," in *2020 4th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India, Oct. 2020, pp. 932–937. doi: 10.1109/I-SMAC49090.2020.9243424.

[27] A. C. Jeba Malar and others, "Automatic number plate recognition system using deep learning algorithms and image processing for surveillance," in *2024 9th International Conference on Science, Technology, Engineering and Mathematics (ICONSTEM)*, Chennai, India, Apr. 2024, pp. 1–5. doi: 10.1109/ICONSTEM60960.2024.10568583.