Indonesian Journal of Engineering, Science and Technology
IJENSET
Volume 01 - Issue 02, December 2024
e-ISSN: xxxx-xxxx; p-ISSN: xxxx-xxxx
Journal Link: https://jurnal.umla.ac.id/index.php/ijenset/index

# VEHICLE NUMBER PLATE CHARACTER RECOGNITION USING OPENCV AND CONVOLUTIONAL NEURAL NETWORK (CNN)

**Bayu Ajie Prasetyo, Eko Handoyo, Hery Ardiansyah***

Corresponding Author Email: hery24@gmail.com

*Study Program of Computer Engineering; Faculty of Science Technology and Education; Universitas Muhammadiyah Lamongan; Lamongan 62218; Indonesia*

**Article Information**

| | | |
|---|---|---|
| Submitted | : | 27th August 2024 |
| Revised | : | 09th December 2024 |
| Accepted | : | 10th December 2024 |
| Paper page | : | 68-78 |
| DOI | : | 10.38040/ijenset.v1i2.1019 |

## ABSTRACT

This research entitled "Vehicle License Plate Character Recognition Using OpenCV and Convolutional Neural Network (CNN)" aims to develop an Automatic Number Plate Recognition (ANPR) system by integrating OpenCV and CNN. The main focus is the application of the You Only Look Once (YOLO) v8 method to detect objects and text in real-time, and the use of EasyOCR to recognize characters. This system is designed to improve the accuracy and efficiency of vehicle license plate recognition. The results of the study showed an average accuracy level of Precision of 40.5%, Recall 100%, and Accuracy 42.16%. These results show that although the model successfully detects all vehicle license plates (with 100% recall), low precision indicates that there are quite a lot of false positives or errors in detection which results in a decrease in the overall accuracy rate.

**Keyword—***Automatic Number Plate Recognition*, *Convolutional Neural Network*, *Deep Learning*, OpenCV, YOLO

## I. INTRODUCTION

The manual recording still common in Indonesia's parking systems, even though the number of vehicles continues to increase, especially in big cities. Crime cases such as selling stolen motorbikes and forging STNK make the owner concern. The Automatic Number Plate Recognition (ANPR) system can overcome this problem by automatically detecting and recording vehicle plate numbers at the parking entrance, as well as assisting agencies in identifying and counting vehicles (Felisa, 2023; Awaimri, 2022).

ANPR (Automatic Number Plate Recognition) is effective for vehicle management in various public places, including parking systems and traffic safety (Chinnaiyan, 2021). Manual recording of vehicle number plates often causes errors, so an automatic device to detect and manage number plate information is needed(Salma, 2021).

The ANPR algorithm involves four main steps: (1) Image acquisition, (2) Number plate detection, (3) Character segmentation, and (4) Character recognition. Currently, Artificial Intelligence (AI) technology is being developed to detect objects and text from images or videos, including on vehicle license plates(Fauzan and Wibowo, 2021).

One of the latest innovations in object detection is the use of deep learning with the You Only Look Once (YOLO) method. YOLO is an object detection approach that works in real-time using a Convolutional Neural Network. In this method, the convolution layer will undergo a convolution process in each network (Hayati, Singasatia, dan Muttaqin, 2023).

Based on this background, a study entitled "Vehicle License Plate Character Recognition Using Open CV and Convolutional Neural Network (CNN)" was conducted with the aim of detecting vehicle license plate objects while being able to detect the license plate text on the vehicle.

## II. METHOD

Convolutional Neural Networks (CNN) is a deep learning algorithm that has a weight sharing feature, which reduces the number of parameters that need to be trained. This feature helps the network to improve generalization ability and reduce the risk of overfitting. In addition, large-scale network implementation becomes easier with CNN compared to other types of neural networks. (Alzubaidi, 2021). For the convolution operation, the basic formula is:

$$FM[i]_{j,k} = \sum m \sum n \; N_{[j-m,k-n]} \, F_{[m,n]} bF \tag{1}$$

Description

*FM[i]* : i-th Feature Map Matrix
*N* : Input image matrix
*F* : Convolution filter matrix
*bF* : Bias value on the filter
*j,k* : Pixel position in the input image matrix
m,n : Pixel position in the convolution filter matrix

Each pixel in the feature map will be processed using the ReLU function, where pixel values less than 0 will be changed to 0, according to the formula *f (x) = max (0, x).* Next, the pooling layer functions to reduce the size of the feature map. The type of pooling applied is max pooling, which selects the maximum value from a certain area in the window. This process is similar to convolution, namely by sliding the window across the image, but this time the window is used to select the maximum value from each specified area. The end result of this process is a feature map matrix consisting of the maximum values that have been selected. As in Figure 1.

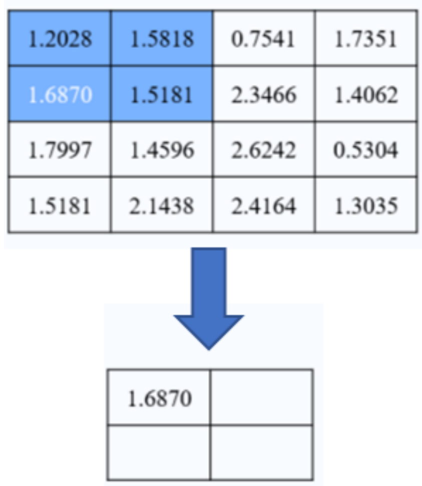


**Figure 1.** Pooling Layer Process

Each neuron is connected to every neuron in the previous layer. The formula for a neuron in a fully connected layer is:

$$y = (\sigma W_x + b) \tag{2}$$

Description:

*W* = weight matrix,
*x* = input vector
*b* = bias
*σ* = activation function (such as sigmoid or ReLU)

To capture global patterns and relationships in input data by connecting each neuron from the previous layer to each neuron in the fully connected layer.

YOLOv8 is one of the object detection algorithms that divides the input image into a grid with a size of SxS. The size of this grid depends on the input size used in the architecture. For YOLOv8, if the input size is 416x416, then the grid sizes are 13x13, 26x26, and 52x52. Each grid cell is tasked with predicting the objects in it, including the bounding box and confidence score that indicate the probability of the object's existence in the bounding box. After the bounding box is identified based on the resulting confidence value, YOLOv8 will predict the class of objects in the bounding box along with their probabilities, thus producing a class probability map (Asyhar, Wibowo, dan Budiman, 2020).

Python is a very popular and feature-rich programming language for Artificial Intelligence development. Many AI developers around the world choose Python because of its various packages, such as TensorFlow, Keras, and Theano, which make it easier for data scientists to develop deep learning algorithms. Python offers excellent support when it comes to implementing deep learning algorithms (Teoh dan Rong 2022). Python can be used for a variety of purposes, including developing web applications, desktop applications, IoT, and many other applications. Python also integrates with database systems and has the ability to read and modify files, making it a popular choice for fast and reliable software prototyping and development. (Rahman., 2023).

Python supports a variety of programming paradigms, including but not limited to object-oriented programming, imperative programming, and functional programming. The Python software logo can be seen in Figure 2. One of Python's strengths is as a dynamic programming language with automatic memory management. Although often used as a scripting language, Python has a broader scope and is applied in various contexts beyond just scripting. Python can be used for various purposes in software development and can be run on various operating system platforms (Muhamad Zulkarnaen, 2024).

**Figure 2.** Python

Figure 2 is a symbol of Python software. NumPy is a library for the Python programming language that provides support for large multidimensional arrays and matrices, and comes with many high-level mathematical functions for processing these data structures. NumPy enables computations with multidimensional array objects and related objects such as masked arrays and matrices. The library offers functions for algebra, basic statistics, random simulation, mathematics, logic, shape manipulation, sorting, selection, I/O, discrete Fourier transforms, and many other applications (Harris . 2020).

When generating arrays, NumPy will default to using the bit depth or color depth depending on the Python environment you are using. If you are working with 64-bit Python, array elements will automatically use 64-bit precision, which requires more memory and may not always be necessary. You can specify the desired bit depth when creating an array by setting the data type (dtype) parameter to int, numpy.float16, numpy.float32, or numpy.float64 (Lawrence, 2019).

Pandas is a library for data analysis and manipulation that supports a variety of data formats, including CSV, TSV, Excel, JSON, and SQL. In general, Pandas provides two main structures for data manipulation: series and DataFrame. Series is a one-dimensional data structure, similar to columns in Excel, while DataFrame is a two-dimensional data structure that takes the form of a table with rows and columns (Palupi, Ihsanto, dan Nugroho 2023).

Pandas helps in organizing the data in .CSV files into various categories like “frame_number”, “track_id”, “car_bbox”, “car_bbox_score”,“license_plate_bbox”,“license_plate_bbox_score”,“license_plate_number” and “license_text_score”. Pandas also supports querying language for extracting information from dataframes and can extract specific columns and/or rows using querying. (Agarwal, 2019) .

The research materials used are vehicle photos and vehicle plates obtained from the roboflow website and directly in the area of the University of Muhammadiyah Lamongan and video recordings as an evaluation at the parking lot and in and out of the University of Muhammadiyah Lamongan area.. as the main material of this study. The device used in vehicle license plate character recognition uses a laptop with AMD Ryzen 5 5600H specifications with Radeon Graphics 3.30 GHz, 16.0 GB. And Jupiter Nootbook as a programming language editor application to create weights and codes in the vehicle license plate character recognition system.

This stage contains research steps from start to finish as shown in Figure 3.

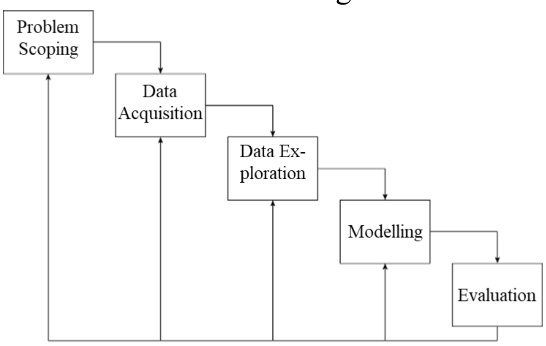


**Figure 3.** Waterfall research procedure

At beginning of the research, researchers must identify the problem as an initial stage. Researchers need to explain in detail the identification of the problem that will be raised in the background of the problem.

1. Observation

In carrying out data collection to complete this research, a common technique is used in scientific activities, namely observation. The observation carried out by the author is a field study of this research which took place at the University of Muhammadiyah Lamongan, the author has obtained data about the location. The observations made by the researcher aim to study the situation and conditions at the location.

2. Literature Study

This research began by conducting a literature study on image processing and consideration of analysis methods using Convolutional Neural Network (CNN). Furthermore, a review of the results of data processing with YOLOv8 and references to creating a detection system using the CNN algorithm were carried out. The research continued with a more in-depth literature study on Optical Character Recognition (OCR) which will be used to scan alphanumeric characters on Motor Vehicle Number Plates.

The next stage is data acquisition, this process is carried out to collect a dataset containing images of motor vehicle plates in the ANPR system, which will be used to train the YOLOv8 model. This study uses photos taken from the Roboflow website.

**Figure 4.** Vehicle plate data

All images are taken in Figure 4 from different angles considering environmental conditions such as lighting, shooting position and rotating the image. Vehicle license plates consist of a subset of 36 characters consisting of 26 characters in the form of letters and 10 characters in the form of numbers. The size of all high-resolution images into fixed-size low-resolution images (416 × 416) using an online web service provided by Roboflow.

The next stage is data exploration, at this stage the dataset is annotated by creating bounding boxes or labeling. The main function of this labeling is to provide the YOLOv8 algorithm with the information needed to know and understand the location and classification of objects in the dataset. The image data obtained is labeled one by one to obtain the coordinates of the ground-truth bounding box which will be compared with the predicted bounding box. By comparing the two ground boxes, the Intersection over Union (IoU) value will be obtained. The labeling process uses source code from the Python language which is then compiled into a labeling application. The results of this labeling will be exported into YOLOv8 with the format *.txt per file. After the entire data set has passed the labeling stage, each data will be divided again into 3, namely train data, valid data, and test data with a total of 238 image annotations in the form of *.txt according to the image file name as in Figure 5.

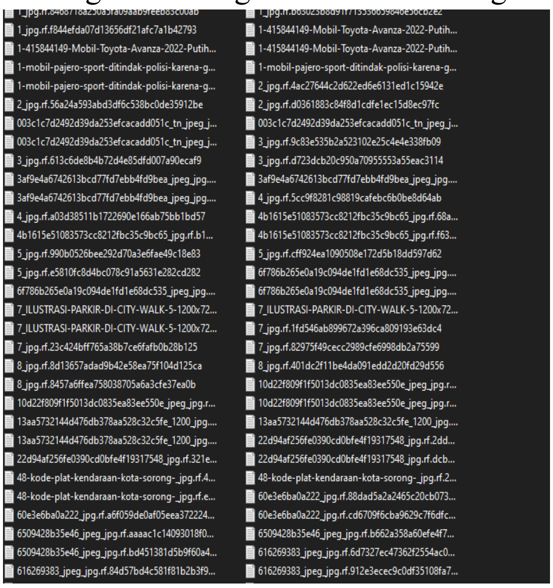

**Figure 5.** Annotation results in *txt format

a. Flowchart train model

Flowchart train model is used as a description of the stages in creating a system. The flowchart display that describes the processing and output generation in the system can be seen in Figure 6.

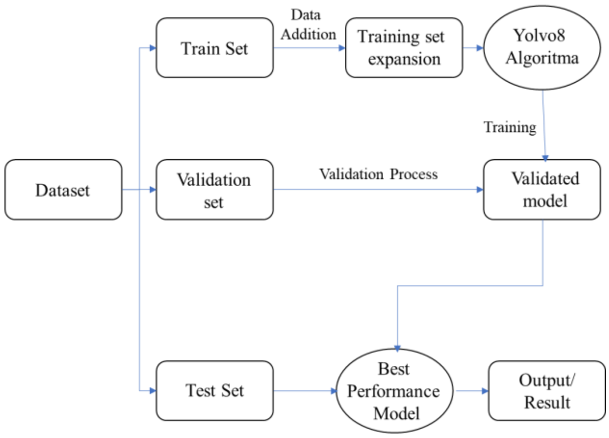


**Figure 6.** Flowchart Training dataset

First, the annotated model is divided into 3, namely: trainset, validset and testset. Trainset is added to expand the training model that will be trained by the YOLOv8 algorithm. Then the validation process is carried out to ensure that the model meets the requirements precisely and accurately by calculating the IoU value between the predicted bounding box and the actual bounding box and calculating the accuracy of object classification from Recall and Precision. After the model is validated, a test is carried out for 100 epochs to see the accuracy value of the model with a time of 1 hour 33 minutes. Then the output produced is the "best.pt" dataset..

b. System Flowchart

After carrying out the model training process, the model testing or evaluation stage is carried out with the flowchart in Figure 7.

In Figure 4, the video to be input is a recording using a smartphone with .mp4 format with a size of 1440 x 1440 pixels. The video pre-processing stage is resized or cut to 640 x 640 pixels to match the input size with the "yolov8n.pt" model dataset. This process ensures that the image remains in a consistent format and in accordance with the dimensions required by the model.

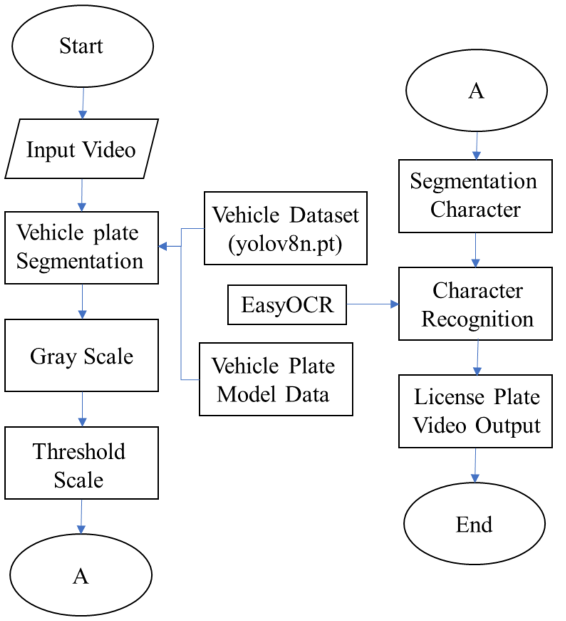


**Figure 7.** System Flowchart

The first process is tracking the vehicle with the function in the YOLOv8 library (.track) to read the frames in the video. Then detect the Region of Interest (RoI) on the vehicle to facilitate the image segmentation process on the OpenCV vehicle plate with the dataset from "yolov8n.pt" for the vehicle. After detecting the RoI of the vehicle, then detect the vehicle plate with OpenCV with data that has been trained in the modeling process with the file name "best.pt" for the vehicle plate. After the license plate image is detected, the gray scale and threshold scale processes are carried out to separate the object from the vehicle plate background based on the difference in brightness or lightness. Furthermore, character segmentation to separate each character on the license plate with OpenCV. After the character segmentation process, the next process is character recognition, to extract text from images with easyOCR because it supports processing images that are natural photos, not necessarily clean images such as in PDF format, videos, and others. So this is suitable for realizing character recognition from license plates where videos from road recordings will be processed each frame to be detected by the Yolov8 model and read by easyOCR for each license plate with the output of a .CSV file. Then the .CSV file is read and inserted back into the video to make the license plate text appear in the video.

c. Use Case Diagram

The Use Case Diagram is used as a graphical representation that can describe the interaction between the user and the system. This User Case Diagram provides an explanation of the functions performed by the system. The Use Case Diagram on the vehicle license plate character recognition system using OpenCV and Convolutional Neural Network (CNN) is presented in Figure 8.

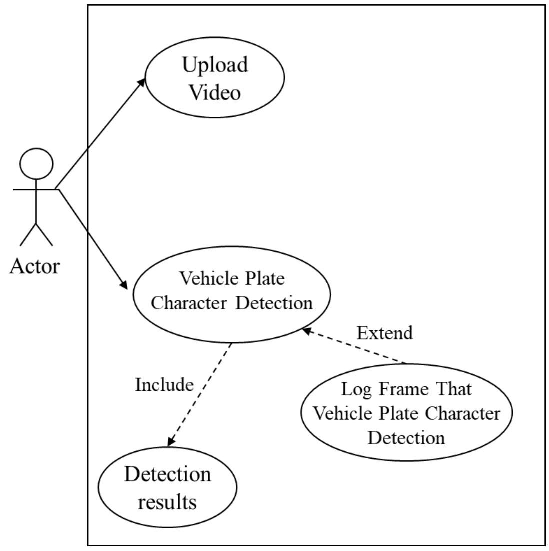


**Figure 8.** use case diagram

In Figure 8, the use case diagram explains the relationship between the actor and the system that has been built. By the actor clicking the "Choose File" button to display the file director where the video file is being tested. Then the actor selects the video to be tested and presses the "open" button, the system displays the name of the selected test data file. Then the actor clicks the "Upload" button, the system detects the vehicle plate character object in each frame of the test video. The log is saved for each frame containing the vehicle plate character object to the system storage in the form of a *CSV format file and displays the detection results in video form. The next stage is Modeling, in this section, the design that has been done is then modeled. The modeling that is done is the implementation of the system interface, selection of test data and detection results.

1. System Interface

In Figure 9, the interface of the human object detection system. This interface displays a button to upload a trial file of vehicle plate character object detection using YOLOv8.


**Figure 9.** System Interface

2. Test Data Selection

As in Figure 10, the system interface has a “Choose File” button which functions to select a video file in the directory file that will be used as test data and an “Upload” button which will send a video file to be processed by YOLOv8 to detect vehicle plate character objects in each video frame sent.

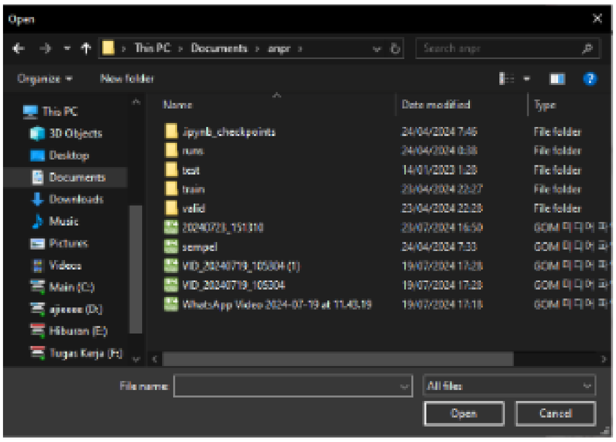


**Figure 10.** Test Data Selection

3. Detection Results

Figure 11 shows the results of vehicle plate object detection on the test data. The detection results are in the form of a bounding box on the human object with a label and confidence value for the detected object.

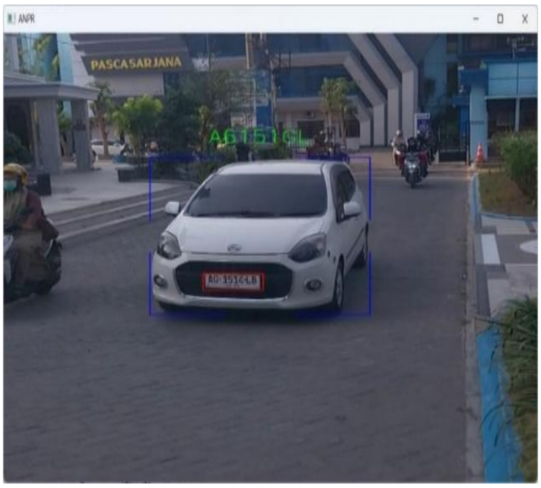


**Figure 11.** Video of detection results on the system interface.

Figure 12 is a file containing vehicle plate objects detected during the detection process in *.CSV format. The results are stored by the system so that they can be used later for the process of investigating the history of the emergence of human objects in the system.

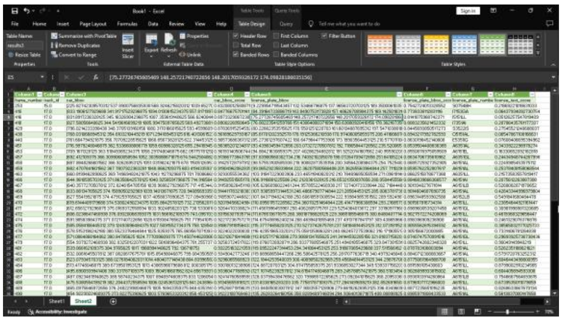

**Figure 12.** Detected vehicle plate object file in *.CSV format.

The evaluation stage involves selecting a model to be used for final output. In this stage, the training data that has been processed with the You Only Look Once (YOLO)v8 algorithm is tested and evaluated. The confusion matrix is used to compare the results of the information provided by the system with the actual information. The evaluation is carried out by measuring metrics such as accuracy, precision, and recall using the confusion matrix. The results of modeling with the YOLOv8 algorithm are presented in the form of a confusion matrix.

**Table 1.** Confusion matrix results

| | Number plate | Background |
|---|---|---|
| **Number plate** | 21 | 3 |
| **Background** | 0 | 0 |

From the confusion matrix results Table 1, confusion matrix performance measurements can be carried out using 4 terms that will show the results of the classification process:

1. True Positive (*TP*) = The number of positive data that is successfully classified correctly by the system.
2. True Negative (*TN*) = The number of negative data that is successfully classified correctly by the system.
3. False Positive (*FP*) = The number of negative data that is incorrectly classified as positive by the system.
4. False Negative (*FN*) = The number of positive data that is incorrectly classified as negative by the system.

Accuracy measures how close the predicted value is to the actual value. The higher the accuracy, the better the performance of the method. The formula for calculating accuracy is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

$$= \frac{21+0}{21+3+0+0}$$

$$= \frac{21}{24}$$

$$= 0{,}875\ (87{,}5\%)$$

Precision measures how accurately the data predicted as positive by the model is compared to the data that is actually positive. Precision is the proportion of True Positive values among all data classified as positive by the model. The formula for calculating precision is as follows:

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$= \frac{21}{21+0}$$

$$= \frac{21}{21}$$

$$= 1\ (100\%)$$

Recall measures how effective the system is in obtaining correct information. The recall calculation formula is as follows.

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

$$= \frac{21}{21+0}$$

$$= \frac{21}{21}$$

$$= 1\ (100\%)$$

## III. RESULT AND DISCUSSION

After the training and evaluation process of the model is complete, the next stage is testing. The following are the results of model testing on video data obtained from recordings using a smartphone with .mp4 format and a resolution of 1440 x 1440 pixels.

From Table 2, the results obtained only detected vehicles and read 6 stable motor vehicle plates on vehicles with track_id 1.0 and 10 objects recognized by the system as vehicles.

**Table 2.** Vehicle plate reading results video 1

| Frame number | Track id | License plate number |
|---|---|---|
| **0** | 1.0 | A0223BH |
| **1** | 1.0 | A0223BH |
| **2** | 1.0 | A0223BH |
| **3** | 1.0 | A0223BH |
| **5** | 1.0 | S4022JB |
| **6** | 1.0 | S4022JB |
| **9** | 1.0 | S4072JB |
| **10** | 3.0 | O5737S9 |
| **15** | 1.0 | S4022JE |
| **18** | 1.0 | S4022JB |

**Table 3.** Results of reading vehicle plate video 2

| Frame number | Track id | License plate number |
|---|---|---|
| **81** | 2.0 | S3422JI |

From table 3, the results obtained only detect vehicles and read motor vehicle plates as many as 1 vehicle plate on a vehicle with track_id 2.0 and 1 other object that the system recognizes as a vehicle.

**Table 4.** Results of reading vehicle plate video 3

| Frame number | Track id | License plate number |
|---|---|---|
| **11** | 4.0 | B2054PK |
| **24** | 4.0 | B2054PK |
| **31** | 4.0 | B2054PK |
| **33** | 4.0 | B2054PK |
| **35** | 4.0 | B2054PK |

From Table 4, the results obtained only detect vehicles and read motor vehicle plates as many as 1 stable vehicle plate on a vehicle with track_id 4.0 and 2 objects that the system recognizes as vehicles.

**Table 5.** Results of reading vehicle plate video 4

| Frame number | Track id | License plate number |
|---|---|---|
| **253** | 15.0 | S0754BH |
| **410** | 17.0 | I7153GA |
| **416** | 17.0 | I5151LL |
| **418** | 17.0 | I7351AI |
| **429** | 17.0 | S3522IS |
| **438** | 17.0 | O5151AL |
| **440** | 17.0 | E5351IG |
| **451** | 17.0 | A6351AL |
| **453** | 17.0 | A6151EL |
| **457** | 17.0 | A6351GL |

From Table 5, the results obtained only detected vehicles and read motor vehicle plates as many as 5 stable vehicle plates on vehicles with track_id 17.0, 2 vehicle plates that were not read by the system as vehicle plates and 7 objects that were recognized by the system as vehicles.

**Table 6.** Results of reading vehicle plate video 5

| Frame number | Track id | License plate number |
|---|---|---|
| **18** | 4.0 | S5655JA |
| **36** | 6.0 | S4363AE |

From Table 6, the test results are obtained which only detect vehicles and read 2 motor vehicle plates and 3 other objects recognized by the system as vehicles.

Meanwhile, Table 7 shows the test results in the form of vehicle detection results and vehicle plate reading using the YOLOv8 method on the test dataset. Testing vehicle detection and vehicle plate reading on the five input data obtained an average Precision of 40.5%, Recall 100%, and Accuracy 42.16%.

**Table 7.** Test results

| Data Input | TP | TN | FP | FN | Precision | Recall | Accuracy |
|---|---|---|---|---|---|---|---|
| **Data Video 1** | 6 | 0 | 10 | 0 | 0,375 | 1,00 | 0,375 |
| **Data Video 2** | 1 | 0 | 1 | 0 | 0,50 | 1,00 | 0,50 |
| **Data Video 3** | 1 | 0 | 2 | 0 | 0,3334 | 1,00 | 0,3334 |
| **Data Video 4** | 5 | 2 | 7 | 0 | 0,4167 | 1,00 | 0,50 |
| **Data Video 5** | 2 | 0 | 3 | 0 | 0,40 | 1,00 | 0,40 |
| | | | Rata-rata | | 0,40502 | 1,00 | 0,42168 |

## IV. CONCLUSION

Based on the results that have been carried out, the following are the conclusions of the study on Vehicle License Plate Character Recognition Using OpenCV and Convolutional Neural Network:

1. The performance of the Vehicle License Plate Classification Model shows excellent results in terms of accuracy. The model achieves an accuracy rate of 87.5%, with precision and recall reaching 100% each after 100 training epochs. This shows that the model is able to identify and classify characters on license plates very accurately and does not make errors in terms of false positives or false negatives.
2. Vehicle Plate Detection and Reading Performance with five input data, the results obtained show an average Precision of 40.5%, Recall of 100%, and Accuracy of 42.16%. These results show that although the model successfully detects all vehicle license plates (with 100% recall), low precision indicates that there are quite a lot of false positives or errors in detection which results in a decrease in the overall accuracy rate.

## ACKNOWLEDGEMENT

Agarwal, V. (2019). Research on Data Preprocessing and Categorization Technique for Smartphone Review Analysis. *International Journal of Computer Applications*, *131*(4), 30–36. https://doi.org/10.5120/ijca2015907309

Al Awaimri, M., Fageeri, S., Moyaid, A., Thron, C., & ALhasanat, A. (2022). Automatic Number Plate Recognition System for Parking. *Studies in Computational Intelligence*, *1006*(November), 155–178. https://doi.org/10.1007/978-3-030-92245-0_8

Asyhar, H. H. Al, Wibowo, S. A., & Budiman, G. (2020). Implementasi Dan Analisis Performansi Metode You Only Look Once (Yolo) Sebagai Sensor Pornografi Pada Video. *EProceedings of Engineering*, *7*(2), 3631.

Fauzan, M. R., & Wibowo, A. P. W. (2021). Pendeteksian Plat Nomor Kendaraan Menggunakan Algoritma You Only Look Once V3 Dan Tesseract. *Jurnal Ilmiah Teknologi Infomasi Terapan*, *8*(1), 57–62. https://doi.org/10.33197/jitter.vol8.iss1.2021.718

Felisa, J., Setiawan, D., & Khalisa, I. (2023). Perancangan Perangkat Lunak Pengenalan Karakter Plat Nomor Kendaraan dengan Metode Convolutional Neural Network. *Media Informatika*, *21*(3), 280–306. https://doi.org/10.37595/mediainfo.v21i3.156

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hayati, N. J., Singasatia, D., & Muttaqin, M. R. (2023). Object Tracking Menggunakan Algoritma You Only Look Once (YOLO)v8 untuk Menghitung Kendaraan. *Komputa : Jurnal Ilmiah Komputer Dan Informatika*, *12*(2), 91–99. https://doi.org/10.34010/komputa.v12i2.10654

Kumar Chinnaiyan, V., Balaji, S., Beny, R., & Kavin Jayasuriya, E. (2021). Automatic Number Plate Recognition System (ANPR): The Implementation. *12th International Conference on Advances in Computing, Control, and Telecommunication Technologies, ACT 2021*, *2021-Augus*(June), 905–908. https://doi.org/10.47524/tjst.21.6

Lawrence, K. (2019). SciPy and NumPy. In *Stuff*. Retrieved from https://www.stuff.co.nz/national/98431311/southern-katipo-helps-defence-force-prepare-for-the-next-major-disaster

Muhamad Zulkarnaen, F. (2024). *Sistem Deteksi Objek Manusia Menggunakan*

*Algoritma Yolov8 Berbasis Kamera Depth Sensor*.

Palupi, L., Ihsanto, E., & Nugroho, F. (2023). Analisis Validasi dan Evaluasi Model Deteksi Objek Varian Jahe Menggunakan Algoritma Yolov5. *Journal of Information System Research (JOSH)*, *5*(1), 234–241. https://doi.org/10.47065/josh.v5i1.4380

Rahman, S., Sembiring, A., Siregar, D., Khair, H., Gusti Prahmana, I., Puspadini, R., & Zen, M. (2023). Python : Dasar Dan Pemrograman Berorientasi Objek. In *Penerbit Tahta Media*.

Salma, Saeed, M., Ur Rahim, R., Gufran Khan, M., Zulfiqar, A., & Bhatti, M. T. (2021). Development of ANPR Framework for Pakistani Vehicle Number Plates Using Object Detection and OCR. *Complexity*, *2021*. https://doi.org/10.1155/2021/5597337

Teoh, T. T., & Rong, Z. (2022). Artificial Intelligence with Python. In *Machine Learning: Foundations, Methodologies, and Applications*. Retrieved from https://link.springer.com/10.1007/978-981-16-8615-3%0Ahttps://doi.org/10.1007/978-981-16-8615-3%0Ahttps://link.springer.com/10.1007/978-981-16-8615-3