**Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)**
Vol. 4, No. 1, March 2021, pp. 11 – 18
p-ISSN: 2614-3372 | e-ISSN: 2614-6150 

# Automatic Car Detection Using Haar Cascade Classifier and Convolutional Neural Network for Traffic Density Estimation

**[1]Miftahul Hasanah, [2] Gulpi Qorik Oktagalu Pratamasunu, [3]Ratri Enggar Pawening**
[1,2,3] Program Studi Informatika, Fakultas Teknik, Universitas Nurul Jadid Probolinggo
Email: [1]hasanahm341@gmail.com, [2]gulpi.qorik@gmail.com, [3]enggar.r@gmail.com

**Article Info**

***Article history:***
Received Nov 17th, 2020
Revised Dec 02nd, 2020
Accepted Dec 29th, 2020

***Keyword:***
Traffic Density Estimation
Haar Cascade
CCTV
CNN

**ABSTRACT**

Based on a survey released by the TomTom Traffic Index in 2018, Indonesia was ranked seventh in the category of the most congested country in the world. One of the factors affecting traffic congestion in Indonesia is an inflexible and conventional traffic management system. Therefore, it is necessary to have a better traffic management system such as a Smart Traffic Light. One way to implement a smart traffic light system is to make a vehicle detection and counting system on the traffic CCTV video automatically. The methods used in this research are Haar Cascade Classifiers and Convolutional Neural Network. Haar Cascade Classifiers have fast computation processes and CNN is applied to validate the detection results of the Haar Cascade method for better accuracy. The average level of accuracy achieved by the system on quiet test data is 82%, normal test data is 69%, and busy test data is 60%. Meanwhile, the average computation time needed by the system for the quiet test data is 0.63 seconds, the normal test data is 0.52 seconds, and the busy test data is 1.05 seconds. Based on the trial results, combining the two methods can increase the accuracy and speed up the computation time required by the system.



***Corresponding Author:***
Miftahul Hasanah,
Program Studi Informatika Fakultas Teknik,
Universitas Nurul Jadid,
1 Jl. KH. Zaini Mun'im Karanganyar Paiton Probolinggo.
Email: hasanahm341@gmail.com

**DOI: http://dx.doi.org/10.24014/ijaidm.v4i1.10785**

## 1. INTRODUCTION

The increasing volume of vehicles number in Indonesia that is increasing rapidly has led to an increase in traffic density thus becoming one of the main factors causing a traffic jam [1]. A traffic jam or vehicle accumulation on the road is the result of transportation sector increase in Indonesia and inadequate road infrastructure development [2]. Apart from being caused by the number of vehicles operating, another cause of the congestion problem has not been resolved until now is a conventional system that is applied to most traffic lights. The current traffic light regulation is considered less flexible because it does not pay attention to the actual road conditions.

The implementation of a better traffic light system management can reduce a traffic jam in Indonesia, one of the ways is by implementing a Smart Traffic Light System. The Smart Traffic Light System is a system that is regulated based on the actual traffic conditions on the road, where the duration of the traffic light is determined based on the queue length of the vehicle at each road junction. The earliest step to implement the Smart Traffic Light System is building a detection and counting system for the number of vehicles in traffic.

Various studies have been conducted to test several object detection methods to find a better method applied for vehicle object recognition in traffic. In the year 2016, research with the title "Simulation and Analysis of Image Processing-Based Smart Traffic Light Systems Digital with Edge Detection and Segmentation Methods" was carried out by [3]. Edge detection is used to determine the boundary of an object before image segmentation processing that can classify an object. The results obtained from this study is a system to determine which segment has the longest queue by calculating the length of centroids on each road

segment. The average accuracy rate of the system reaches 80.93%. However, the accuracy level will decrease when there are non-vehicles objects such as shadows, trees whose branches are above the road, and people who are crossing the road are caught by the system because the system will consider these objects as vehicles.

This research will focus on the recognition and determination of vehicle objects number on each road segment by utilizing Closed Circuit Television (CCTV) traffic video data installed in every intersection. The method used in this research are Haar Cascade Classifier and Convolutional Neural Network (CNN). Merging the Haar Cascade method with CNN is done because CNN takes a long time to detect objects in real-time. Also, the Haar Cascade method is used to retrieve bounding boxes from vehicles before they can be detected by CNN. The advantage of the Haar Cascade Classifier method is a very fast computation process because this method does not depend on each pixel value of an image but the number of pixels in a square [4]. Meanwhile, the CNN method is applied to improve the detection results of the Haar Cascade Classifier filter so that it can achieve a better level of accuracy. Therefore, with a very fast computation process and a good level of accuracy, the combination of Haar Cascade Classifier and CNN methods are considered compatible to be implemented in vehicle detection real-time systems.

## 2. RESEARCH METHOD

In this research, the method used is the combination of the Haar Cascade Classifier and Convolutional Neural Network methods to detect and count car objects on CCTV videos. This research will be conducted in several stages which have been summarized in research models such as Figure 1. The research model is designed to facilitate the process of identifying and estimating the number of vehicles detected by the system.

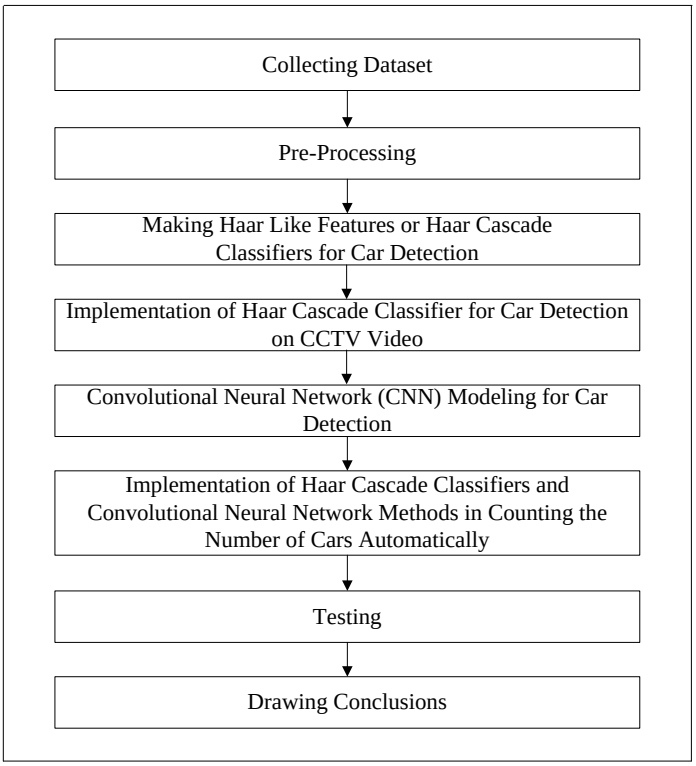


**Figure 1.** Research Model

### 2.1. Collecting Dataset

In this study, the process of collecting data on car images and non-car images is not carried out independently but by utilizing a secondary dataset that is widely distributed in studies domestic and abroad. The car dataset used is The Comprehensive Cars (CompCars) Dataset shared by [5]. The dataset is a collection of thousands of single-car images from various sides, types, backgrounds, and circumstances.

From the original dataset which consists of various sides of the car images, the image used is only the front view of the car as the dataset in this study. In addition choosing a front view car image, this study also only selects and uses car images in daytime condition as an initial stage of research. Total car object images taken from the CompCars dataset in this study are 1000 images.

Then this study also takes traffic CCTV video data from the youtube page with the search keywords "CCTV traffic", "roads", "cars", and "highway". Total videos obtained from the page youtube amount to 12 videos with file format *.mp4. The videos will then be extracted and used as an additional dataset.


**Figure 2.** Car and Non-Car Dataset on Haar Cascade Filter Making

Meanwhile, for non-car images, the dataset used is a dataset of background images shared by [6]. Dataset is a collection of background images with several categories such as bed, living room, kitchen, sub-urban, and so on. This study uses 1000 image backgrounds of the dataset which does not have the car object therein that has been selected manually. An example of the collected dataset can be seen in Figure 2.

### 2.2. Preprocessing

At this stage, the dataset preprocessing will be done before the model training process. This stage aims to adjust the image format to make the Haar Cascade Classifiers for car detection. The original image on the CompCars dataset is converted into *.bmp format. The process of reformatting the image use the Paint app on Windows 10.

### 2.3. Making Haar Like Features or Haar Cascade Classifiers for Car Detection

Tool and tutorial that is used as references for making Haar Like Features or Haar Cascade Classifiers to detect cars are conducted by [7]. The process starts from collecting images, sequencing negative images or background images, marking objects in positive images or car images, creating vectors from positive images, the Haar Training process, the formation of an XML file from the training results. Most of the processes at this stage are done by running the batch file from [7].

### 2.4. Implementation of Haar Cascade Classifier for Car Detection on CCTV Video

At this stage, the Haar Cascade Classifiers filter that has been successfully created will be implemented to detect car objects on CCTV video. After being tested, the result of the system detection still has a lot of false-positive value, which are still non-car objects detected as cars. Hence, on this stage, the video extraction process will be carried out to obtain a dataset of car and non-car images based on the detection results of the Haar Cascade Classifiers filter which will then be used in making Convolutional Neural Network (CNN) model. After the video extraction process is complete, the extracted image results will be manually selected to get the good image needed. An example of image extraction results can be seen in Figure 3.

### 2.5. Making Convolutional Neural Network (CNN) Model for Car Detection

The CNN model is a model to classify an image and predict whether the image is a car or not. In the CNN architecture, an input image is detected its attributes/features by using 3 filter convolutions. After the convolution process, max pooling will produce 3 convolutional images with a smaller resolution. At last, the max-pooling results is included into a hidden layer Multi-Layer Perceptron (MLP). The Architecture Model is in Figure 4. Model created with Python programming language, Tensorflow, and Keras library. The training data to use is collected image data in the previous stage. Data that has been manually selected will be grouped into two classes, namely the car class and the non-car class. The total training data used is 3000 images, with details of 1500 car images and 1500 non-car images. Some examples of cars and non-cars dataset used can be seen in Figure 3.

Figure 3. Car and Non-Car Dataset on CNN Model Making

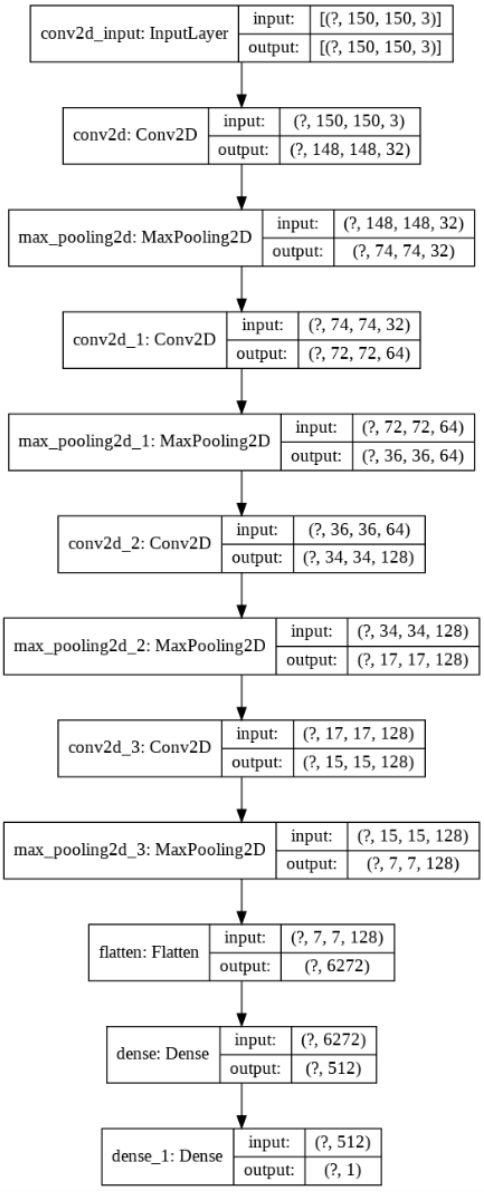


**Figure 4.** CNN Architecture Model

### 2.6. Implementation of Haar Cascade Classifiers and Convolutional Neural Network Methods in Counting the Number of Cars Automatically

This phase is the stage of programming to detect and count the number of car objects detected by combining the Haar Cascade Classifiers filter and the previously created Convolutional Neural Network model. The program is compiled using the Python programming language version 3.6.8 and runs on the Miniconda 4.8.0 platform. Libraries needed in the preparation of this program include OpenCV python, NumPy, TensorFlow, and Keras.

The first detection is done by the Haar Cascade Classifiers filter, the training data will be checked as a whole whether there are car objects in it. Then any detected car objects will be trimmed and saved into a temporary file with *.jpg format. The program will load the detected Haar Cascade Classifiers detection results, then it will be re-validated by the CNN model whether the detected object is a car or not. After going through the validation process with the CNN model, the position of objects detected in the training data will be marked with a red square box. Then the program will count the number of squares as a representation of traffic density.

### 2.7. Testing

The testing process is done on test data with three traffic conditions, that are quiet traffic, normal traffic, and busy traffic. The test data used is image data extracted from traffic CCTV videos. The images are then categorized based on ongoing traffic conditions. Each condition is tested with 10 images with *.jpg format so that the total test data used in this study amounted to 30 images in total.

In the trial process, the method success rate is measured based on computation time and rate accuracy. Besides being tested with the method understudy, this phase will also compare the program trial results with other deep learning methods such as You Only Look Once (YOLO) and Single Shot Detection (SSD). The YOLO program to be compared refers to research conducted by [8]. Meanwhile, the SSD program is based on research by [9].This matter is done to find a better and more suitable method for real-time detection and counting of car objects. The test data example in quiet, normal, and crowded conditions seen in Figure 5.

Figure 5. Test Data in Quiet, Normal, and Busy Conditions

### 2.8. Drawing Conclusion

After doing the entire series of processes in the research model, at this stage, an analysis will be done on the results of the trial methods used as well as comparison results between the methods used and other methods. So that it can be concluded the advantages and disadvantages of each method in detecting and calculating car objects in real-time.

## 3. RESULT AND ANALYSIS

At this stage, the experimental data will be analyzed to determine the success rate of the method. Two discussion points will be presented. Those are the results analysis of Haar Cascade Classifiers and Convolutional Neural Network (CNN) method trial on quiet, normal, busy test data and results analysis of a comparative trial of the Haar Cascade Classifiers and CNN methods with the YOLO and SSD methods.

### 3.1. Haar Cascade Classifiers and Convolutional Neural Network (CNN) Method.

Based on Table 1, the highest average accuracy rate is in the quiet test data category, which is 82%. Then the average level of accuracy decreased in normal test data with a percentage of 69% and busy test data with the lowest average accuracy rate of 60%. On the busy test data, the average system accuracy rate is the lowest because there are too many car objects in one image, so these objects tend to accumulate at certain points which causes the system does not recognize the object shape properly. Besides, the method can only detect private car type vehicles so that other vehicle objects such as buses, trucks, and other large vehicles will

not be detected properly by the system. It is because the dataset used for model training is a type of private vehicle only. Then another reason that affects the detection results is the distance of the camera from the object. Objects that are located far enough from the camera cannot be recognized properly by the system.

Apart from the level of accuracy, the computation time required by the system to detect objects is also analyzed. The average computation time on the busy test data is 1.05 seconds, which tends to be longer than the average computation time on the normal test data which is only 0.52 seconds and the quiet test data is 0.63 seconds. The system requires a longer computation time to detect busy test data because if there are more objects detected by the Haar Cascade Classifier filter, the more objects will have to be validated by the CNN method. The CNN method requires a longer computation time than the Haar Cascade Classifier method because the CNN method still checks the objects detected by the Haar Cascade method one by one. Therefore, the computation time required by the system depends on the number of objects contained in the test data.

**Table 1.** Accuracy Level and Computation Time of Haar Cascade and CNN Methods

| No | Accuracy Level | | | Computation Time (second) | | |
|---|---|---|---|---|---|---|
| | Quiet | Normal | Busy | Quiet | Normal | Busy |
| 1 | 100% | 80% | 55% | 0.98 | 0.72 | 1.14 |
| 2 | 100% | 100% | 70% | 0.61 | 0.67 | 1 |
| 3 | 67% | 80% | 80% | 0.79 | 0.75 | 1.05 |
| 4 | 50% | 43% | 70% | 0.50 | 0.37 | 1.04 |
| 5 | 50% | 43% | 43% | 0.32 | 0.38 | 0.8 |
| 6 | 100% | 50% | 53% | 0.45 | 0.37 | 1.2 |
| 7 | 100% | 60% | 53% | 0.67 | 0.44 | 1.25 |
| 8 | 50% | 75% | 57% | 0.49 | 0.42 | 1.05 |
| 9 | 100% | 80% | 66% | 0.43 | 0.52 | 1.14 |
| 10 | 100% | 80% | 50% | 1.03 | 0.54 | 0.82 |
| Average | 82% | 69% | 60% | 0.63 | 0.52 | 1.05 |

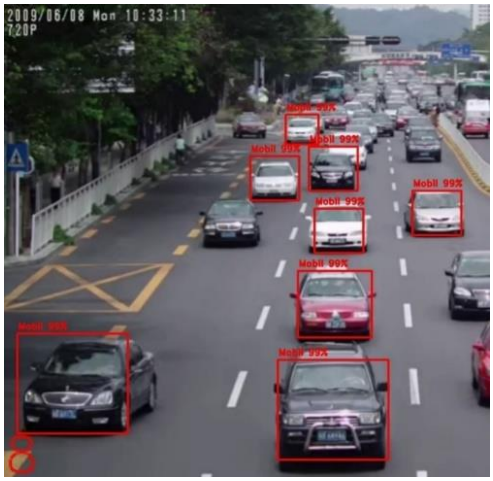


**Figure 6**. Detection Results of Haar Cascade and CNN

The output of this method is an image with the marking of the detected object, the level of accuracy, and the estimated number of vehicles. Figure 6 is an example of the detection results using the Haar Cascade Classifiers and Convolutional Neural Network methods.

### 3.2. Haar Cascade Classifiers and Convolutional Neural Network (CNN), You Only Look Once (YOLO), and Single Shot Detection (SSD) methods

With the same test data, the Haar Cascade and CNN methods are also compared with the YOLO and SSD methods. The test data used were 10 traffic images with normal categories. Figure 6 is an example of the comparison of the Haar Cascade Classifiers and CNN methods with YOLO and SSD. The three methods are compared based on the level of accuracy and computation time required. The trial results can be seen in table 2.

Based on the results of the method trials in Table 2, it can be concluded that the highest average accuracy rate is achieved by the SSD method with a percentage of 74%, then followed by the Haar Cascade and CNN methods with an insignificant difference of 69%, and the YOLO method with an average accuracy rate of 20%. Whereas in the computation time achievement required by each method, SSD requires the longest

average time compared to the other two methods, that is 4.87 seconds, YOLO method for 1.91 seconds, and Haar Cascade and CNN methods require an average time for 0.52 seconds.

It can be concluded that although the average accuracy rate of the SSD method is the highest compared to the other two methods, the computation time spent is the longest. So that the SSD method is not suitable for implementation in real-time detection systems. On the other hand, Haar Cascade and CNN methods require a very short time to detect an image and the resulting average accuracy is also good enough, so that it is more suitable to be implemented.

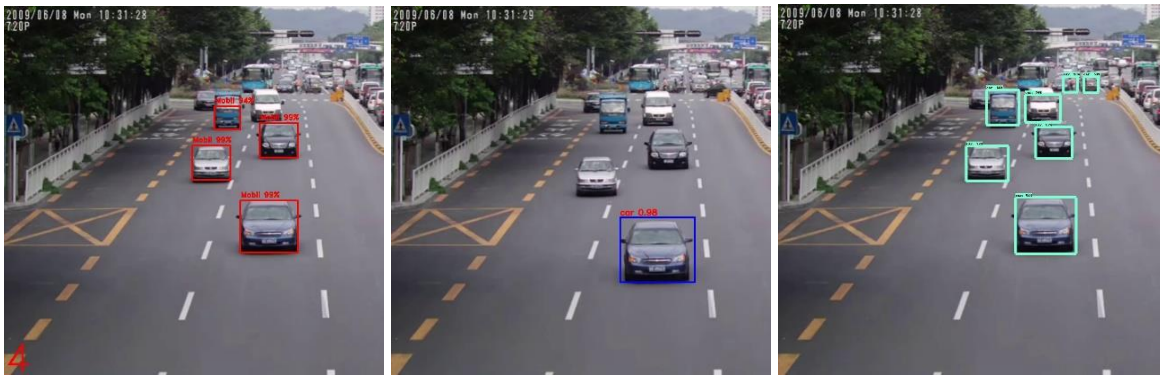

**Figure 7.** Test Results of the Haar Cascade Classifiers and CNN, YOLO, and SSD methods

**Table 2.** Accuracy Level and Computation Time of Haar Cascade Method and CNN, YOLO, and SSD

| No | Accuracy Level | | | Computation Time (second) | | |
|---|---|---|---|---|---|---|
| | Haar Cascade and CNN | YOLO | SSD | Haar Cascade and CNN | YOLO | SSD |
| 1 | 80% | 0% | 100% | 0.72 | 4.42 | 3.81 |
| 2 | 100% | 20% | 80% | 0.67 | 1.65 | 4.31 |
| 3 | 80% | 20% | 60% | 0.75 | 1.64 | 4.54 |
| 4 | 43% | 0% | 86% | 0.37 | 1.64 | 4.77 |
| 5 | 43% | 28% | 43% | 0.38 | 1.63 | 4.98 |
| 6 | 50% | 0% | 83% | 0.37 | 1.63 | 5.20 |
| 7 | 60% | 20% | 100% | 0.44 | 1.63 | 5.42 |
| 8 | 75% | 50% | 50% | 0.42 | 1.64 | 5.67 |
| 9 | 80% | 20% | 80% | 0.52 | 1.64 | 5.93 |
| 10 | 80% | 40% | 60% | 0.54 | 1.63 | 4.06 |
| Average | 69% | 20% | 74% | 0.52 | 1.91 | 4.87 |

## 4. CONCLUSION

Based on the results of testing the methods that have been developed, it can be concluded that the Haar Cascade and CNN methods can detect and estimate the number of vehicles well. The level of accuracy and computation time required depends on the number of objects in the test data. The fewer objects contained in the image, the higher the level of accuracy, and the faster the computation time is spent. Whereas in the comparison of the Haar Cascade and CNN methods with the YOLO and SSD methods, the Haar Cascade and CNN methods are the most suitable methods to be implemented because of their fast computation time and fairly good accuracy.

## REFERENCES

[1] G. L. Djavendra, S. Aisyah and E. R. Jamzuri, "Desain Sistem Pengatur Lampu Lalu Lintas Dengan Identifikasi Kepadatan Kendaraan Menggunakan Metode Subtraction," *Jurnal Nasional Teknik Elektro*, pp. 130-137, 2018.

[2] R. Adistya and . M. A. Muslim, "Deteksi dan Klasifikasi Kendaraan menggunakan Algoritma Backpropagation dan Sobel," *Journal of Mechanical Engineering and Mechatronics*, pp. 65-73, 2016.

[3] D. A. Priutomo, R. Magdalena and N. Andini, "Simulasi Dan Analisis Sistem Smart Traffic Light Berbasis Pengolahan Citra Digital Dengan Metode Deteksi Tepi Dan Segmentasi," *e-Proceeding of Engineering*, pp. 478-485, 2016.

[4] M. Syarif and W. , "Deteksi Kedipan Mata Dengan Haar Cascade Classifier Dan Contour Untuk Password Login Sistem," *Techno.COM* , pp. 242-249, 2015.

[5] L. Yang, P. Luo, C. C. Loy and X. Tang, "A Large-Scale Car Dataset for Fine-Grained Categorization and Verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[6] L. Fei-Fei and P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[7] M. Rezaei, "Creating a Cascade of Haar-Like Classifiers: Step by Step," pp. 1-8, 2013.

[8] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[9] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Fathi, A. Korattikara, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CVPR* , 2017.

**BIBLIOGRAPHY OF AUTHORS**

Miftahul Hasanah. The author is student of S1 Teknik Informatika, Universitas Nurul Jadid Probolinggo.

Gulpi Qorik Oktagalu Pratamasunu, S.Pd, M.Kom. The author is a lecturer of Teknik Informatika Universitas Nurul Jadid Probolinggo.

Ratri Enggar Pawening, M.Kom. The author is a lecturer of Teknik Informatika Universitas Nurul Jadid Probolinggo.