Available *online* at : http://jnte.ft.unand.ac.id/

Jurnal Nasional Teknik Elektro

| ISSN (Print) 2302-2949 | ISSN (Online) 2407-7267 |

An Embedded Convolutional Neural Network for Maze Classification and Navigation

Gunawan Dewantoro, Dinar Rahmat Hadiyanto, Andreas Ardian Febrianto

Department of Electronic and Computer Engineering, Satya Wacana Christian University, Salatiga, 50711 Indonesia

ARTICLE INFORMATION

Received: April 07, 2023
Revised: July 07, 2023
Accepted: July 07, 2023
Available online: July 31, 2023

KEYWORDS

Convolutional Neural Network, Maze, Classification, Navigation, Robot

CORRESPONDENCE

Phone: +62 857 4343 8874

E-mail: gunawan.dewantoro@uksw.edu

A B S T R A C T

Traditionally, the maze solving robots employ ultrasonic sensors to detect the maze walls around the robot. The robot is able to transverse along the maze omnidirectionally measured depth. However, this approach only perceives the presence of the objects without recognizing the type of these objects. Therefore, computer vision has become more popular for classification purpose in robot applications. In this study, a maze solving robot is equipped with a camera to recognize the types of obstacles in a maze. The types of obstacles are classified as: intersection, dead end, T junction, finish zone, start zone, straight path, T–junction, left turn, and right turn. Convolutional neural network, consisting of four convolution layers, three pooling layers, and three fully-connected layers, is employed to train the robot using a total of 24,000 images to recognize the obstacles. Jetson Nano development kit is used to implement the trained model and navigate the robot. The results show an average training accuracy of 82% with a training time of 30 minutes 15 seconds. As for the testing, the lowest accuracy is 90% for the T-junction with the computational time being 500 milliseconds for each frame. Therefore, the convolutional neural network is adequate to serve as classifier and navigate a maze solving robot.

INTRODUCTION

Over the last few decades, the interest in deep neural network has soared among researchers as it is able to tackle with a large amount of data. The applications of deep neural network encompass a range of fields by creating various artificial intelligence models, one of which is an image detection engine using a deep convolutional neural network (CNN). Basically, deep convolutional neural network works by extracting features in digital images. These key features of an image characterize a specific object that can be used to classify and discriminate against other objects. Several pre-processing stages such as image normalization and image segmentation are essential for the feature extraction process. The result of feature extraction is subsequently used for image classification or detection [1].

Deep neural networks play a variety of roles in many fields. In the health sector, deep convolutional neural networks are used to detect lungs disease [2], detect mask users as a pre-emptive measure against the spread of Covid-19 [3][4], and to recycle plastic waste such as nails and screws [3]. In the automotive field, deep convolutional neural networks are applied to recognize traffic signs and also responsible in the navigation of autonomous cars [4] - [6]. In addition, other applications such as human facial expression detection [7] and face detection for security can also take advantage of deep convolutional neural network [8] - [12].

In robotic applications, neural network is widely used for maze mapping application [13] - [16]. As for the robot navigation, maze robots traditionally employ ultrasonic sensors to detect the circumstances around the robot - [17] and apply the right-handed algorithm to determine the robot's motion [18]. However, ultrasonic sensors has shortcomings as it can not recognize the type of objects ahead; therefore, difficult to visualize the grand picture of the maze. This study attempts to bridge that gap by replacing the distant-measuring approach with image processing method using a camera. An artificial neural network is used to help robots navigate in a maze. The dataset was captured directly using OpenCV with the amount of 24,000 images data. By utilizing convolution operations in digital image data, the robot can learn from the training data to recognize the types of obstacles being faced by the robot. Convolutional neural network extracts the characteristics of digital images through a multitude of hidden layers that are built so as to find a network model that is considered appropriate. Once the trained model is obtained, the robot is not only able to navigate but able to recognize the types of obstacles in the maze. Subsequently, this paper is organized as follow: section 2 explains the research method used in this study, section 3 presents the results and discusses the key findings and limitations, and section 4 concludes the work.

https://doi.org/10.25077/jnte.v12n2.1091.2023

METHOD

Experimental Stages

Figure 1 shows the workflow used in this study. First, a webcam is used to capture the video frames and create the dataset, with each frame in the video recording being taken separately until collecting 3000 image data for each class. The capture is governed by Jetson Nano by means of program listings in the OpenCV library. The image dataset is grouped into 8 classes, namely intersection, dead end, T junction, finish zone, start zone, straight path, T–junction, left turn, and right turn. Each class contains 2500 training data, 430 validation data, and 70 test data. In digital image pre-processing, the original digital image is rescaled from 1080×720 pixels to 224×84 pixels. Subsequently, the image is converted from multiple channels to single channel by generating the grayscale images. Then, the collected image data are randomly selected and grouped as training data and validation data which will then be used for training in convolutional neural networks. The network model is composed by a multitude of hidden layers and functions. Some parameters within the network are finely tuned to yield the best outcome i.e., accuracy. Finally, the trained network model is tested to see if the robot can correctly classify the obstacles.

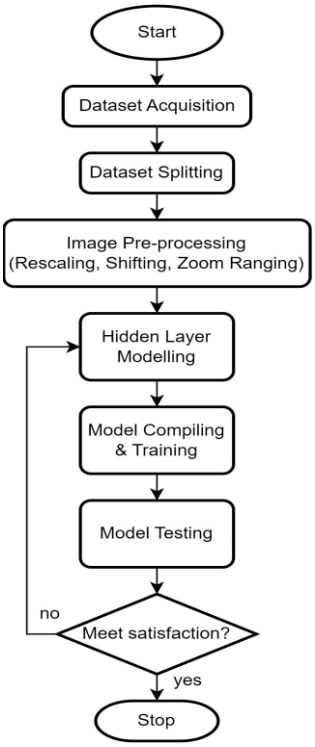

Figure 1. Research Workflow

The performance assessment is carried out by running the robot along the maze. The network model is embedded in the Nvidia Jetson Nano developer kit and equipped with a 720p30fps webcam. The test is carried out using 70 images for each class which are not previously used for training and validation. The detection results are then used for assessing the correctness of the robot's motion. The test results are expected that the robot is able to classify with an accuracy of 80% and move correctly based on the classification results.

Hardware Setup

The robot is constructed in three levels and equipped with two wheels for the locomotion drives. The robot is able to traverse according to the image classifier program. It was built in the form of 3-level robot car having two active wheels and one passive wheel. Figure 2 shows the schematic diagram of all hardware interconnections. All commands are processed in the Nvidia Jetson Nano developer kit which has a Quad-core ARM A57 processor with a clock speed of 1.43 GHz, a 128-core Maxwell GPU, RAM of 4 GB, and internal storage of 64 GB. In addition, the robot is equipped with a 1080p Logitech C92 webcam, gearbox, Li-Po batteries, L298N motor drivers, and voltage regulators.

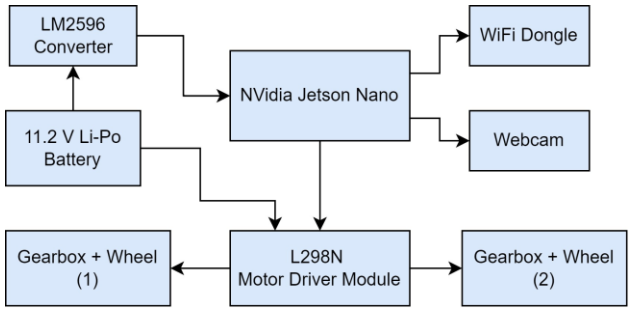

Figure 2. Schematic Diagram

Jetson Nano developer kit is employed to perform the image classification because of its high level graphics processing unit [19]. The outcome of this classification is used to determine the robot's motion using Python3 programming. In this study, the task of the convolutional neural network is to classify the circumstances to which the robot is approaching. These circumstances are divided into eight classes, namely the intersection, dead end, finish zone, start zone, straight path, T-junction, left turn, and right turn. The image classification process begins when the robot is turned on and start capturing the image. The predetermined initial position is always located in one spot, i.e. the starting zone. Then, the convolutional neural network classifies the obstacles with respect to the captured image in front of robot. Having received the classification results, the robot determines the motion of each wheel to steer the robot accordingly, as shown in Table 1. Since the robot is driven by means of a pair of active wheels, then the locomotion is based on the differential drive method. For example, if the robot turns to the left, then the left motor must stop while the right motor moves forward. Similarly, if the robot rotates clockwise, then the left motor must move forward while the right motor moves backward. In this study, the robot applies the right-hand rule algorithm, meaning that it prioritizes turning right whenever it encounters two or more alternatives to pass through, such as intersections and T-junctions. In addition, if the robot encounters a dead end, it will rotate clockwise instead of move backward in order the webcam for capturing another class category. The robot repeats the commands and keeps moving until it reaches the finish zone and stops.

To conduct the experiment, a 120cm×240cm maze was built from 1-cm thick plywood with white walls and black floor, as depicted in Figure 3. The maze comprises of eight types of obstacles that will be learned by the robot as mentioned above. The design of

the maze was inspired by the realistic road, with white markers that can be a feature used in the neural network.

Table 1. Instruction Table

Classification Result	Motor Output	
	Left	***Right***
Intersection	Forward	Stop
Dead End	Forward	Backward
Finish Zone	Stop	Stop
Start Zone	Forward	Backward
Straight Path	Forward	Forward
T-Junction	Forward	Stop
Left Turn	Stop	Forward
Right Turn	Forward	Stop

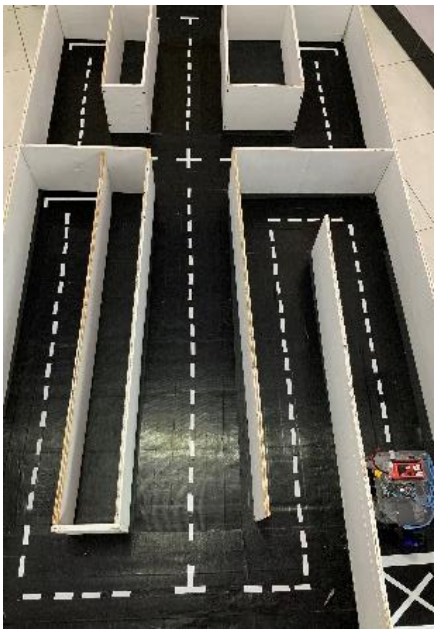

Figure 3. The Maze

Data Acquisition

The data was acquired using a 1080p webcam and OpenCV library. The resulting images are in grayscale with dimension of 224×84×1 pixel. The complete dataset is as much as 24,000 and divided into 8 classes for each training, validation, and testing data, as shown in Table 2. The training and validation data were used together during the network learning stage, while the testing data were used to test the trained network. These testing data are not a subset of the training data nor validation data.

Figure 4 shows the sample of image data for each class used for training and validation. The variation of images used in model training is constituted from a range of image point of views or different angle perspective captured by the camera. These images might be taken from the same video recording but at different sampling instants, some of which are blurry. These blurred images were still kept in dataset to train network even in the presence of distorted input to better infer under various circumstances. We can also notice in Figure 5 that the paths along the maze are marked with white stripes to help classify the circumstances.

Table 2. The Number of Samples of Each Class

No.	Class	Number of Samples		
		Training	***Validation***	***Testing***
1	Intersection	2500	430	70
2	Dead End	2500	430	70
3	Finish Zone	2500	430	70
4	Start Zone	2500	430	70
5	Straight Path	2500	430	70
6	T-Junction	2500	430	70
7	Left Turn	2500	430	70
8	Right Turn	2500	430	70

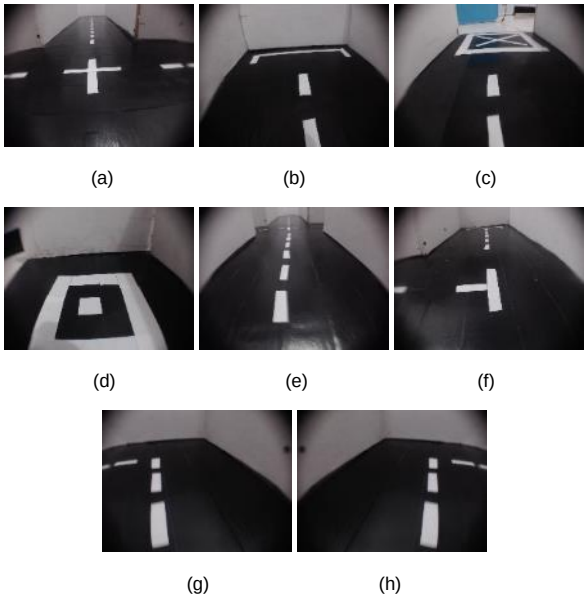

Figure 4. Dataset Images: (a) Intersection, (b) Dead End, (c) Finish Zone, (d) Start Zone, (e) Straight Path, (f) T-Junction, (g) Left Turn, (h) Right Turn

Convolutional Neural Network Modelling

Prior to the training phase, the image is split into training and validation data. At this stage, the data is randomly split using features in the Tensorflow framework-Keras API with a ratio of 85:15. This split ratio provides 2,500 and 430 data for training and validation, respectively. Then, these split data proceed with the augmentation process, which includes rescaling, width shifting, height shifting, and zoom range adjusting. These augmentations aim to make the data more flexible during training.

Table 3. Convolutional Neural Network Architecture

No.	Layer	Output Shape
1	Convolution	(None, 222, 82, 6)
2	Average Pooling	(None, 111, 41, 6)
3	Convolution	(None, 109, 39, 16)
4	Average Pooling	(None, 54, 19, 16)
5	Convolution	(None, 52, 17, 32)
6	Convolution	(None, 50, 15, 32)
7	Average Pooling	(None, 25, 7, 32)
8	Flatten	(None, 5600)
9	Dense	(None, 120)
10	Dense	(None, 84)
11	Dropout	(None, 84)
12	Dense	(None, 8)

https://doi.org/10.25077/jnte.v12n2.1091.2023

The convolutional neural network architecture was selected by adopting several previously used architectures such as LeNet, AlexNet, ZF Net [20][21]. The first layer is convolution layer with 224×84 grayscale image data input. This layer uses a kernel size of 3×3, 16 channels, and a rectified linear unit (ReLu) activation function. The second and third layer employ the same layer model as before, and then followed by maxpooling layers with a filter size or kernel size of 2×2. The subsequent three layers use the same model with 32 channels. For the next three layers, 64 channels are used, and then a flatten layer is utilized to convert the data dimension to 1×28800. The final layer is a fully-connected layer that will be used for prediction and classification. The full architecture of the convolutional neural network is shown in Table 3. The model training process uses a total of 1,370,676 parameters, equipped with Adam function as the optimization algorithm. In this study, the loss function is the cross-entropy as given in Equation 1.

$$\text{Loss} = -\sum_{i=1}^{n} t_i \log(p_i) \quad (1)$$

where t_i is the target label and p_i is the softmax probability for class i. This network architecture is chosen by heuristic approach, meaning that the selection is purely based on intuition and trial-and-error practices.

RESULTS AND DISCUSSION

To demonstrate the performance, a number of indicators are used to assess the results. First, the training and validation accuracy are measured during the model learning stage. Then the trained network model is tested by applying a testing dataset not previously used during the training stage. The size of testing data is 70 images for each class. The result of model testing is represented in the form of an evaluation matrix. Finally, an experiment is done by demonstrating the maze solving robot in a real maze.

Model Training

The network model training is based on the hidden layer architecture as explained in Table 3. The training stage was carried out on a laptop with Intel i-7 processor, GPU RTX 3050, and 16 GB RAM. The training process run iteratively, with 30 epochs being the upper bound of the training period. The value of the accuracy of the training results and the validation of each epoch is directly proportional to the quality of the model. Meanwhile, the loss indicates how well the model fits the training data or new data. In this study, we take advantage of the early stopping feature in keras.callbacks with a patience value of 6. This aims to take the best value and terminate the training process if the training result does not improve after the next 6 steps. This is also effective to avoid network overfitting [22]-[24].

Table 4. Model Training Process

Epoch	Accuracy		Loss	
	Train.	*Val.*	*Train.*	*Val*
1	0.3058	0.5384	1.7311	1.1263
2	0.5078	0.5253	1.2420	1.0416
3	0.5814	0.6599	1.0440	0.9691
4	0.6243	0.7994	0.9452	0.7203
5	0.6683	0.9119	0.8470	0.5213
6	0.7061	0.9087	0.7715	0.5029
7	0.7345	0.9174	0.7046	0.3578
8	0.7522	0.8128	0.6608	0.5025
9	0.7742	0.8788	0.6139	0.3414
10	0.7878	0.9276	0.5777	0.2810
11	0.8035	0.8407	0.5367	0.4696
12	0.7213	0.9061	0.4995	0.2676
13	0.8241	0.9052	0.4781	0.2504
14	0.8339	0.8273	0.4591	0.5457
15	0.8467	0.8724	0.4229	0.3141
16	0.8560	0.8799	0.3993	0.3069
17	0.8636	0.8294	0.3817	0.5430
18	0.8724	0.8029	0.3534	0.7257
19	0.8758	0.8666	0.3523	0.3760

Table 4 implies that the best model training is at epoch 13 (shaded), with a training accuracy of 82.41% and validation accuracy of 90.52%. This model is the best model because the validation loss is the parameter required for the early stopping feature by keras.callbacks. If the validation loss does not improve for the next 6 epochs, then the training process terminates. As we can see in Table 4, the validation loss of epoch 14 – 19 still greater than that of epoch 13. As a result, the training process stops at epoch 19 even though the maximum epoch is 30.

Model Testing

Upon completion of the training, the trained network was then tested with a set of data not previously used in the training stage. This to evaluate the extent of generalization of the trained network. The confusion matrix in Figure 5 states the quality of the model that allows visualization of the performance of the Convolutional Neural Network. The number of testing data for each class is 70, giving a total of 560 testing dataset. The quality of model is in accordance with the accuracy results in Table 4. The result representation is composed based on the confusion matrix variant 3 as explained in [25].

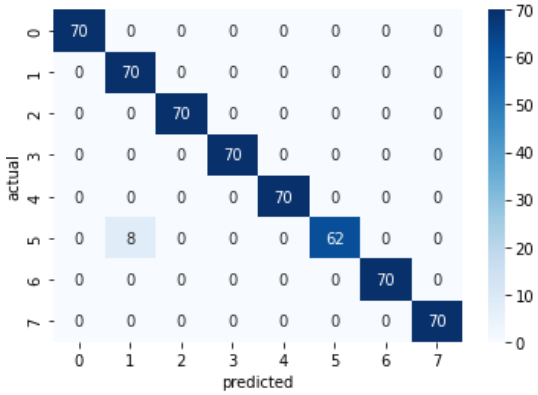

Figure 5. Confusion Matrix

In details, labels 0-7 are the representations of the classes. Label 0 represents the intersection class, with 70 correctly predicted data. Label 1 represents the dead-end class, with 70 correctly predicted data. Label 2 represents the finish zone, with 70 correctly predicted data. Label 3 representing the start zone class, with 70 correctly predicted data. Label 4 represents the straight path class, with 70 correctly predicted data. Label 5 represents the T-junction class, with 62 correctly predicted data. The remaining 8 incorrect prediction recognize the T-junction as a dead end,

giving an accuracy of 88.57%. This misclassification is mainly due to the similarity of the color distribution of the captured images between the two. As shown in Figure 4(b) and (f), the black parts have somewhat same shape and distribution on the image. So, in case of the webcam angle being tilted, this misclassification may occur. This Label 6 represents the left turn class, with 70 correctly predicted data. Lastly, label 7 represents the right turn class, with 70 correctly predicted data. Based on the results mentioned above, this model is sufficient to solve the maze despite being the first research employing convolutional neural network to help solve the maze. This affirms that the convolutional neural network is a powerful tool in image classification tasks; therefore, contribute its popularity in robotic applications [26].

Table 5 shows the precision, recall, and F1-score which is in agreement with the training results in Table 4. The metrics are formulated using Equation (2) – (4).

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{3}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

where TP denotes True Positive, FP denotes False Positive, and FN denotes False Negative. These metrics becomes a reference for concluding the quality of the model. For example, a precision of 1 means that a False Positive is not found for that class. In determining the motion of the robot while solving the maze, the False Negative value becomes critical as it indicates the robot's incapability to undertake the appropriate commands. As in the condition that the robot must do a U-Turn when it encounters a dead end, the robot may not do a U-Turn and cause failure in solving the maze if the number of False Negative is considerably high [27]. Throughout the classification process, the computational time is 50 ms for each video images frame.

Table 5. Model Report

Class	Precision	Recall	F1-Score
Intersection	1.00	1.00	1.00
Dead End	0.90	1.00	0.95
Finish Zone	1.00	1.00	1.00
Start Zone	1.00	1.00	1.00
Straight Path	1.00	1.00	1.00
T-Junction	1.00	0.89	0.94
Left Turn	1.00	1.00	1.00
Right Turn	1.00	1.00	1.00

Maze Solver Experiment

The robot, with a Jetson Nano developer kit, runs the CNN model that is used to determine the motion of each wheel. The experiment was done twice, where the first was conducted off the maze, while the latter was conducted on the maze. The first test aims to evaluate the motor motion in response to the classification result. To do so, the robot was tested off the maze and the classification results were given by a specific hard-coded program with a range of recorded images, meaning that the classes inferred by the robot was not from the instantaneous webcam images. The robot's response to the classification results gives the appropriate motor motion, and the success rate is sought using Equation (5).

$$\text{Success Rate} = \frac{\text{Succesful response}}{\text{Total trials}} \times 100\% \tag{5}$$

Table 6 shows that the motor motions are in accordance with the results of the classification carried out by the model with an overall success rate of 100%, with 10 trials for each class. These success rates of 100% are achieved because the CNN model is able to classify the previously-stored images in the Jetson Nano. For example, if the robot detects an intersection image, then the robot responds by rotating the left motor forward and stopping the right motor, as explained in [28].

Table 6. Motor Motion Results

Classification Result	Motor Output		Success Rate (%)
	Left	*Right*	
Intersection	Forward	Stop	100
Dead End	Forward	Backward	100
Finish Zone	Stop	Stop	100
Start Zone	Forward	Backward	100
Straight Path	Forward	Forward	100
T-Junction	Forward	Stop	100
Left Turn	Stop	Forward	100
Right Turn	Forward	Stop	100

However, the first testing of CNN model on robot was carried out indirectly by using the recorded video images. So, the second test was conducted on the maze with the real-time image capturing from the webcam. It turns out that the robot moved wiggly along the given trajectory and failed to accomplish the navigation tasks. This is because the video images reading process using Python on the Jetson Nano developer kit experienced a decrease in the frame rate to as low as 2 frames per second, which causes the robot's motion does not work properly in real-time capture and cause the robot not able to fully navigate along the maze. In light of these limitations, a number of measures can be considered to overcome this problem, such as:

1. Hardware selection, e.g. use a mini-computer that supports parallel processing to execute vector operation faster.
2. AI model selection, e.g. employ network architecture that comprises fewer number of layers in the fully-connecting to reduce computational load without sacrificing generalization.
3. Parameter optimization, e.g. choose the proper frame rate that works well with the given development kit.

CONCLUSIONS

Convolutional neural network works properly for a maze solving robot to classify a variety of obstacles. The architecture of the convolutional neural network plays a critical role in obtaining a satisfactory accuracy. In this study, the selected architecture consists of four convolution layers, three pooling layers, and three fully-connected layers. The results show that accuracy reaches 82.41% for training and 90.52% for validation. Meanwhile, the

https://doi.org/10.25077/jnte.v12n2.1091.2023

testing accuracy shows that all classes are correctly predicted, except for T-junction with accuracy of 88.57%. If tested using recorded images, the robot's motors can move in accordance with the results of the classification carried out by the model with an overall success rate of 100%. However, in the real-time maze solving experiment, the video images reading process is deteriorated due to the frame rate decline which leads to some delays in the real-time image capture. Instead of Nano, a minicomputer can be further explored for this application to overcome the hardware limitations.

ACKNOWLEDGMENT

The authors would like to thank Satya Wacana Christian University for supporting this research under research grant No. 189/Pen./Rek./6/V/2021.

REFERENCES

[1] M. Jogin, Mohana, M. S. Madhulika, G. D. Divya, R. K. Meghana and S. Apoorva, "Feature Extraction Using Convolution Neural Networks (CNN) and Deep Learning," in 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2018, pp. 2319-2323, doi: 10.1109/RTEICT42901.2018.9012507.

[2] H. C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," IEEE Transactions On Medical Imaging, vol. 35, No.5, pp. 1-15, May 2016.

[3] Z. Wang, H. Li, X. Zhang, "Construction Waste Recycling Robot For Nails And Screws: Computer Vision Technology And Neural Network Approach", Automation in Construction, vol. 97, pp. 220-228, Hongkong, 2019, ISSN 0926-5805, https://doi.org/10.1016/j.autcon.2018.11.009.

[4] D. A. Alghmgham, G. Latif, J. Alghazo, and L. Alzubaidi, "Autonomous Traffic Sign (ATSR) Detection and Recognition Using Deep CNN," in Procedia Computer Science, vol. 163, pp. 266-274, 2019. ISSN 1877-0509, https://doi.org/10.1016/j.procs.2019.12.108.

[5] B. Ko, H. J. Choi, C. Hong, J. H. Kim, O. C. Kwon, and C. D. Yoo, "Neural Network-Based Autonomous Navigation For A Homecare Mobile Robot," in IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 403–406, Jeju, 2017, doi: 10.1109/BIGCOMP.2017.7881744.

[6] Kocić, Jelena, N. Jovičić, and V. Drndarević. "An End-To-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms." Sensors, vol. 19, no. 9, 2019.

[7] D. L. Z. Astuti and Samsuryadi "Kajian Pengenalan Ekspresi Wajah Menggunakan Metode PCA Dan CNN," in Prosiding Annual Research, vol. 4, no. 1, pp. 293-297, 2018.

[8] A. Chavda, J. Dsouza, S. Badgujar and A. Damani, "Multi-Stage CNN Architecture for Face Mask Detection," in 6th International Conference for Convergence in Technology (I2CT), Maharashtra, pp. 1-8, 2021. doi: 10.1109/I2CT51068.2021.9418207.

[9] A. Ulhaq, J. Born, A. Khan, D. P. S. Gomes, S. Chakraborty and M. Paul, "COVID-19 Control by Computer Vision Approaches: A Survey," IEEE Access, vol. 8, pp. 179437-179456, 2020, doi: 10.1109/ACCESS.2020.3027685.

[10] Almabdy, Soad, and Lamiaa Elrefaei. "Deep Convolutional Neural Network-Based Approaches for Face Recognition," Applied Sciences, vol. 9, no. 20, pp. 1-21, 2019.

[11] Permana, D. Ajie. "Pendeteksi Wajah Bermasker Menggunakan Metode Faster R-CNN," Dissertation Universitas Komputer Indonesia, 2021.

[12] Li, Yang, et al. "Face Recognition Based on Recurrent Regression Neural Network." Neurocomputing, vol. 297, pp. 50-58, 2018.

[13] A. Zarkasi, H. Ubaya, C. D. Amanda, and R. Firsandaya, "Implementation of RAM Based Neural Networks On Maze Mapping Algorithms for Wall Follower Robot," Journal of Physics: Conference Series, vol. 1196, no. 1, pp. 1-6, 2019, doi: 10.1088/1742-6596/1196/1/012043.

[14] A. Rodriguez-Tirado, D. Magallan-Ramirez, J. D. Martinez-Aguilar, C. Francisco Moreno-Garcia, D. Balderas and E. Lopez-Caudana, "A Pipeline Framework for Robot Maze Navigation Using Computer Vision, Path Planning and Communication Protocols," 2020 13th International Conference on Developments in eSystems Engineering (DeSE), pp. 152-157, 2020. doi: 10.1109/DeSE51703.2020.9450731.

[15] Rostami, S. M. Hosseini, et al. "Obstacle Avoidance of Mobile Robots Using Modified Artificial Potential Field Algorithm," EURASIP Journal on Wireless Communications and Networking, vol. 70, pp. 1-19, 2019.

[16] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," Proceedings. 1985 IEEE International Conference on Robotics and Automation, pp. 500-505, 1985, doi: 10.1109/ROBOT.1985.1087247.

[17] S. Suryanarayana, V. Akhila, "Autonomous Maze Solving Robot Using Arduino", International Journal of Advanced Research in Engineering and Technology (IJARET), vol. 12, no. 3, pp. 595-603, 2021, doi: 10.3421/IJARET.12.3.2021.054

[18] A. Sabril and N. M. Abdal, "Perbandingan Waktu Tempuh Mobile Robot Dalam Arena Labirin Dengan Algoritma Tangan Kiri Dan Algoritma Tangan Kanan," Jurnal Media Elektrik, vol. 17, no. 3, 2020. p-ISSN: 1907-1728, e-ISSN: 2721-9100.

[19] A. A. Süzen, B. Duman, and B. Şen, "Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN", International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, 2020, doi: 10.1109/HORA49412.2020.9152915

[20] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., "Imagenet Large Scale Visual Recognition Challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.

[22] S. Salman and X. Liu, "Overfitting Mechanism and Avoidance In Deep Neural Networks," arXiv preprint 2019, arXiv: 1901.06566.
[23] Q. Xu, M. Zhang, Z. Gu, "Overfitting Remedy by Sparsifying Regularization on Fully-Connected Layers of CNNs," Neurocomputing, vol. 328, pp. 69-74, 2019, doi: https://doi.org/10.1016/j.neucom.2018.03.080.
[24] X. Ying, "An Overview of Overfitting and its Solutions," Journal of Physics: Conference Series, vol. 1168, no. 2, 2022.
[25] Z. Guoping, "On the confusion matrix in credit scoring and its analytical properties," Communications in Statistics - Theory and Methods, vol 49, no. 9, 2020. https://doi.org/10.1080/03610926.2019.1568485
[26] R. Wassem and W. Zenghui, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," Neural Computation, vol. 29, no. 9, 2017.
[27] S. Ahmad, S. U. Ansari, U. Haider, K. Javed, J. U. Rahman, and S. Anwar, "Confusion matrix-based modularity induction into pretrained CNN," Multimedia Tools and Applications, vol. 81, pp. 23311 – 23337, 2022. https://doi.org/10.1007/s11042-022-12331-2
[28] S. Konduri, E. O. C. Torres, P. R. Pagilla, "Dynamics and Control of a Differential Drive Robot With Wheel Slip: Application to Coordination of Multiple Robots," Journal of Dynamic Systems, Measurement, and Control, vol. 139, no. 1, 2017.

 https://doi.org/10.25077/jnte.v12n2.1091.2023