FOREX Publication
Open Access | Rapid and quality publishing

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X

Advanced Artificial Neural Network for Steering and Braking Control of Autonomous Electric Vehicle

Eka Nuryanto Budisusila[1*], Sri Arttini Dwi Prasetyowati[2], Bustanul Arifin[3], Muhammad Khosyi'in[4] Agus Adhi Nugroho[5] and Muhamad Haddin[6]

[1,2,3,4,5,6]*Universitas Islam Sultan Agung;* ekanbs@unissula.ac.id[1] *arttini@unissula.ac.id[2], bustanul@unissula.ac.id[3], chosyi@unissula.ac.id [4] , agusadhi@unissula.ac.id [5] , haddin@unissula.ac.id [6]*

***Correspondence:** Eka Nuryanto Budisusila; ekanbs@unissula.ac.id

ABSTRACT- Sensors are necessary for an autonomous electric vehicle (AEV) system to identify its environment and take appropriate action, such avoiding obstacles and crashes. Despite their limitations about color, light, and non-metallic items, cameras, radar, and lidar are widely employed to detect objects surrounding a vehicle. Ultrasonic sensors are weather and light-resistant. Thus, the goal of this work was to create object detectors by combining multiple long-range ultrasonic sensors into a multi-sensor circuit. The Arduino processor incorporates an artificial neural network that uses the advanced artificial neural network as a novel approach to control the sensors. There are two steps to this method: offline training and implementation test. The most ideal neural network weights are found offline using the adaptive back propagation algorithm, and the best fixed weight is then embedded into the neural network software on Arduino for implementation test. Because the system can sense more detail about the vehicle's surroundings and accurately avoid obstacles, the definition of actions by taking the object's distance into consideration is better. As a result, the training can produce an output that is closed to the target with 0.001 errors.

Keywords: electric vehicle; AEV; ANN; ultrasonic.

ARTICLE INFORMATION

Author(s): Eka Nuryanto Budisusila, Sri Arttini Dwi Prasetyowati, Bustanul Arifin, Muhammad Khosyi'in, Agus Adhi Nugroho and Muhamad Haddin;

Received: 24/06/2024; **Accepted**: 26/07/2024; **Published**: 10/08/2024;
e-ISSN: 2347-470X;
Paper Id: IJEER 2406-17;
Citation: 10.37391/ijeer.120318
Webpage-link:
https://ijeer.forexjournal.co.in/archive/volume-12/ijeer-120318.html

Publisher's Note: FOREX Publication stays neutral with regard to Jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

The land transportation industry, which encompasses a broad spectrum of vehicles, is expanding at a rapid pace. There are vehicles with two, three, four, or more wheels [1]. On numerous roadways, these cars are becoming more and more crowded, creating traffic congestion. Regretfully, a number of accidents have been caused by these growths, most of which are the consequence of human error. According to the World Health Organization (WHO), road accidents result in the deaths of up to 1.3 million people annually, making it one of the primary causes of death worldwide [2]. Although a lot of technology has been developed to guard against the likelihood of mishaps, there is still a lack of technology to really prevent these accidents. Capable technology is needed to do this in order to prevent accidents before they happen, such as an early warning system that is semi-automated [3], or automated systems that are directly linked to the car's system. When it's necessary, the car can automatically steer itself into a safer direction, apply the brakes, and sound the horn [4]–[6]. Accident risk can therefore be further decreased and, in certain situations, eliminated [7], [8].

The study suggests maximizing the usage of ultrasonic sensors to expand their potential applications beyond vehicle parking guidance systems, considering the issues. This advice is based on the advancement of ultrasonic sensor technology, which is becoming more and more sophisticated and is something to think about using as an object detection sensor surrounding the car to prevent accidents brought on by collisions with objects.

This led to the selection of a long-range ultrasonic sensor with enhanced object detecting capabilities. Long-range ultrasonic sensors have a detection range that allows them to find things up to ten meters away. Given that ultrasonic sensors are very reliable and unaffected by color or lighting, their use can supplement and improve existing sensors like lidar, radar, and cameras. Additionally, the ultrasonic sensor can detect a wide range of items, both metal and non-metal, and is resistant to weather conditions like rain [9], [10]. Ultrasonic sensors are assembled into multi-sensors to be able to detect objects on all sides of the vehicle [11]. Eight ultrasonic sensors are employed in this study. Each of the front, left, right, and back sides has two sensors. For the front and back, long-range sensors that can detect objects up to ten meters away are employed. The long-range ultrasonic sensor, which has a maximum range of five meters, is utilized for the sensors on the right and left side. By doing this, the distance to the front and rear sensors is intended to be increased based on the movement of the car and the likelihood that another vehicle may be following it [12].

A microcontroller with an inbuilt artificial intelligence system is used to process the signal from the ultrasonic sensors. This study uses an advanced artificial neural network with both

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X

offline and implementation test. To be able to operate properly, an artificial neural network requires the weights of each network, such as input, hidden layer, and output. These weights can be obtained by offline training the network [13]–[17]. During training, the network system's output will be compared to a predefined target. The system will change the current weights if the output deviates from the aim. The weight change system can be adjusted in several ways. In this work, the adaptive weight adjustment system makes use of the back-propagation algorithm. Reviewing and readjusting the current weights in reverse, from the output to the input, is what the back-pro algorithm does. The training process is then repeated until the output value approaches or equals zero, has a minimum error value, and is as near to the goal value as feasible [18]–[20]. The neural network system in the Arduino Due microcontroller, which is coupled to input and output devices, is implemented using the preset weight values once the offline training is finished and the ideal weights have been determined. An ultrasonic multi-sensor circuit with a variety of input data is used as the input device, and a dot matrix display circuit is used as the output device to show the target activities. Nine actions, selected from five neural network outputs, will be shown to the system's output in this study. The vehicle's actions in avoiding obstructions, like braking, applying gentle pressure, turning, and moving forward in a straight line, are the targets. The turning left and right actions are expanded to six actions by considering to the detection range of the sensors. The detection range is divided to near, medium, and far. And the expanded actions are narrow turning left, medium turning left, wide turning left, narrow turning right, medium turning right, and wide turning right. This study is also related to our previous study in lateral driving control [21].

The novelties of the study are: the use of ultrasonic sensors as multi-sensor circuits with robustness to lighting condition and the ability to detect metal or non-metal objects instead of cameras, lidar and radar, and the ANN method enriched with predefined actions as a new method to make neural network systems run quickly on implementation test.

The advantages of using the method are that the training of neural network can be conducted offline with required iteration and minimum error value, and the obtained weight values of the network can be set as fixed-weights and implemented in online training system with various data of input taken from the output signal of the sensors. Because there are no iterations in the online training, this method allows for faster completion.

2. MATERIALS AND METHODS

The four primary components of the system's construction are its inputs, neural networks, outputs, and actions. The inputs section explains the different kinds of sensors that are utilized, how they are positioned inside a multi-sensor circuit, and how they work together to detect objects and prevent collisions. The section on neural networks explains how to create an artificial neural network that will be trained as a deep learning system. It also covers the parameters that are supplied to the network and the training procedure [22], [23]. The system's responses to the inputs that the network has processed are covered in the outputs section. Furthermore, by taking the detection range into account, the action section expands upon the outputs. The neural network configuration with the input layer, hidden layer, and output layer in the neural network section, as well as the output layer, and the system action with its "one hot encoding" codes in the output section are all shown in *figure 1*, along with the sensor circuit and its locations in the input section [24].

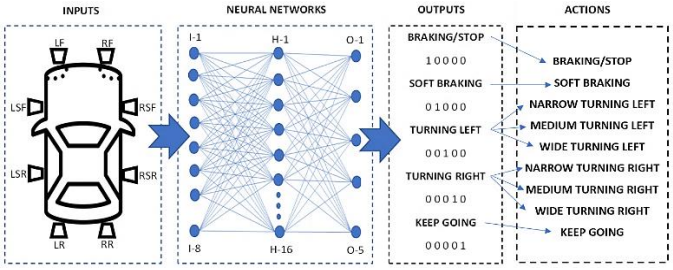
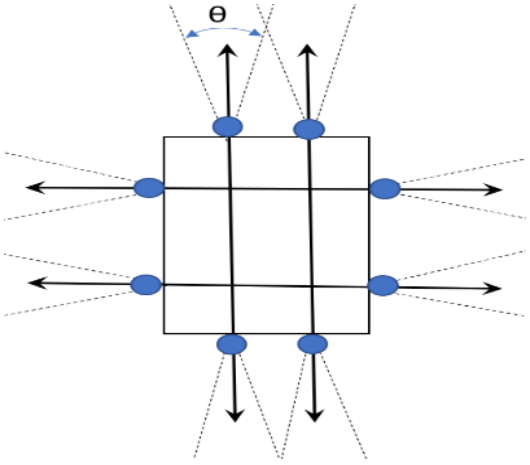


Figure 1. The design of the system

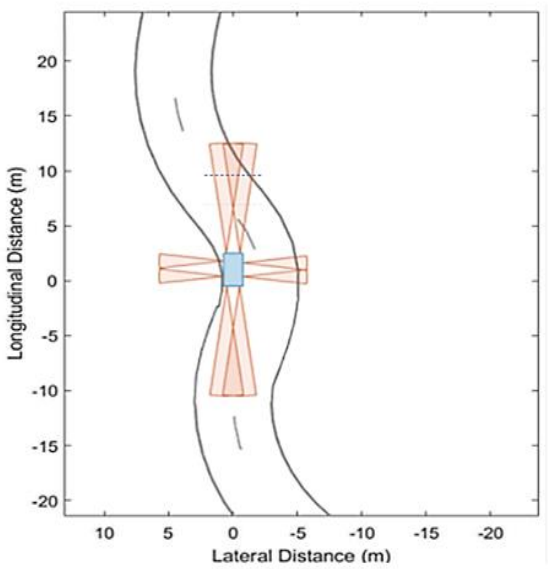
2.1. Inputs Section

Eight ultrasonic sensors, one on each side of the car, are used in this study. The car has four sensors total—two on each side of the left and right sides, two in the front, two in the back, and two in between. With a detection range of up to 10 meters, the Maxbotix MB7383HRXL sensor is fitted on both the front and rear sides. Additionally, the Maxbotix MB7380HRXL sensor which have a maximum detecting range of five meters, are utilized for the left and right side [25]. The goal of each type is to fulfill the requirement of detection; the front and rear sides require lengthier detection times since the vehicle moves forward at different rates and there is a chance that another vehicle may follow it at the same pace. Its purpose is to keep a safe gap between the cars if the leading vehicle brakes unexpectedly. Furthermore, a range of maximum range detection based on vehicle speed can be achieved by adjusting the input voltage of these sensors. While there is no need for regulation, the sensors on the left and right sides are set to fixed range detection. Every sensor has beam angle θ of 10 degrees, which *figure 2 (a)* depicts the sensor configuration.



(a)

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X



(b)

Figure 2. (a) The sensors positioning, headings, and angles (b) The W track scenario

To create a multi-sensor circuit, all sensors are connected to an Arduino Due processor, which processes the signal from sensor units. Every sensor is connected to one selected pin of Arduino, allowing the sensors to be processed individually and determine whether an object is detected on each side. Each sensor sends the detected range information to the processor, if there is no object detected, the sensor sends the maxi-mum range information, shorter ranges information indicates that there is an object in front of sensor. It is important to know that the minimum detection range of all sensors is approximately 50 centimeters, so objects within this range can't be detected accurately [25].

The goal of the sensors' positioning scenario is to identify things on both sides of the vehicle by detecting their surroundings. The system is programmed to halt vehicle movement when any object is spotted by the front sensors, noting that no items were identified by the back sensors. If a car collides with another vehicle behind it, the system is intended to apply mild braking before coming to a stop. The system directed the car to turn left or right if the left-front (LF) or right-front (RF) front sensors were the only ones to detect the object. The car is told to turn right if an object is detected in the left front, and vice versa, if there are no objects detected on the left or right side of the car. If not, the car is programmed to stop or gently brake. Additionally, even though all other sensors notice the object, the system will permit the car to proceed forward if neither of the two front sensors picks up any information.

Driving situations must adjust to the complicated settings with several bends, such the C- or W-shaped tracks shown in *figure 2(b)*. The vehicle's speed will be lowered by the system, along with the voltage supply to the sensor. This will shorten the sensor's ideal distance. This modification complies with the features of the sensor, which has three detecting range levels depending on voltage. Accurate object detection is the goal of the adaptation, regardless of whether the object is in the path or not.

Eight input layer neurons, sixteen hidden layer neurons, and five output layer neurons make up the neural network architecture used in this work. The signal from the sensor output that has been processed by the processor is received by the neurons in the input layer. The inputs are normalized to binary before being processed by the neural network, where a value of "0" denotes no object detection input and a value of "1" denotes object detection input. The input that falls inside the ideal range line is identified as the detected ones, and this completes the classification procedure.

Each input neuron has a weight value that indicates the strength of its propagation, and these links connect to the neurons in the hidden layer. 16 neurons are arranged in the hidden layer such that the number of inputs is doubled. Every hidden neuron in the network is given the label bias once one input is supplied. Usually, the bias has a fixed value of -1 or 1. Additionally, it is made for the output layer's hidden layer. With connection weights and biases, every neuron in the hidden layer is connected to every neuron in the output layer.

This neural network's five output neurons are set up according to how the system acts. We will compare these five outputs to the goals. If there are discrepancies between the outputs and the targets, the system will use the back-propagation technique to reevaluate the weights in each network link.

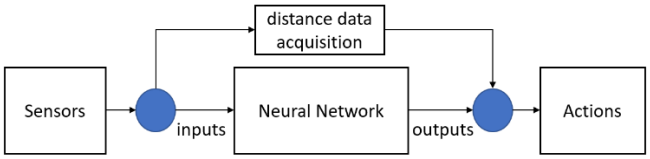
2.2. Output Section

The five actions that result from the outputs are braking or stopping, applying gentle pressure to the brakes, turning left, turning right, and moving forward. When the sensors identify an object, these activities are performed in response to the input condition. The action statements should be transformed to numerical values so that the system can initialize them. The action statements, which are 10000 for braking, 01000 for soft braking, 00100 for turning left, 00010 for turning right, and 00001 for moving forward, can be changed to logical or numerical values by employing the "one hot encoding" method.

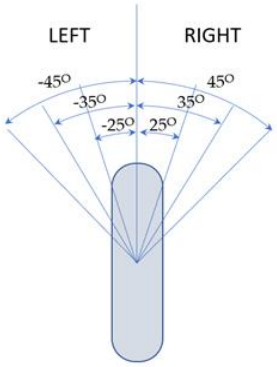
2.3. Action Section

The outputs of turning left and right are expanded in the action part to include narrow, medium, and wide turns as well as narrow, medium, and wide turns in the action section. These actions are conducted by considering to detection range data taken from front sensor signal. The actions represent the wheel turning angle when the system meets the object in front by determining its range to the vehicle in near, medium, or far. The far category is the range between 601 cm to 900 cm, the medium is 301 cm to 600 cm, and the near is 50 cm to 300 cm. The far detected object will affect to the narrow turning angle of the wheel, the medium detected object affects to the medium turning angle, and the near detected object affects to the wide turning angle. The turning angles are set to 25 degrees for the narrow, 35 degrees for the medium, and 45 degrees for the wide. The aim of the actions is to avoid obstacle as soon as possible precisely. Detail diagram is shown in *figure 3*.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X



(a)



(b)

Figure 3. (a) The distance data acquisition for the actions (b) The turning angle

2.4. Training Method

There are two types of system training processes: offline training and implementation test. The implementation test of system is conducted after offline training done. The network is trained to determine the optimal network link weights with the least amount of error divergence between targets and outputs through offline training. The implementation test ensures that the neural network outputs correspond to the predefined actions by applying the optimal network weights to the vehicle system.

An artificial neural network with a back-propagation algorithm is used in the adaptive process of establishing the best network weights. The weights are reviewed and changed adaptively based on deviations from targets and outputs. The weights are saved as neural network fixed-weights once the ideal weights have been determined. After that, the artificial neural network is tested using these fixed-weights in an online training session before being integrated into the vehicle system. Real-time processing and quicker processing of online training are made possible by this technology. *Figure 4* depicts the training technique.

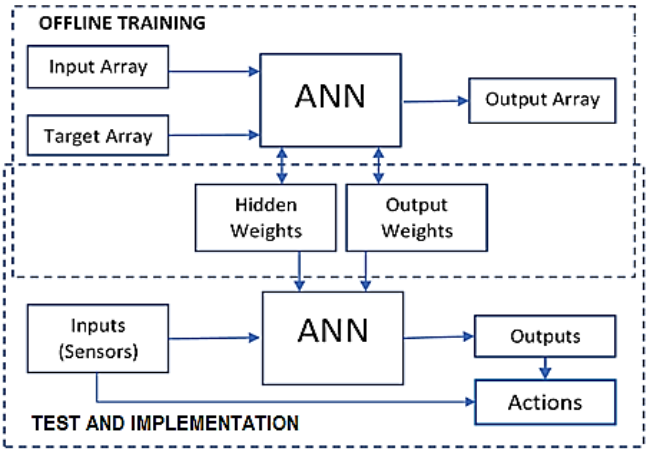


Figure 4. The training method

In contrast to offline training, online training adds actions to the artificial neural network's output. The output of the training results with fixed weights is reformulated by considering the left and right front sensor detection distance data. The action to be performed, which is the vehicle's wheel turning angle, will depend on the detecting distance. The wider turning angle indicates the closer object. The goal of this activity is to precisely and swiftly avoid the impediment item.

2.5. The Offline Training

Several inputs are trained to the network by supplying the targets during the offline training process. The number of sensors mounted on the car determines how the inputs are configured. The network has eight inputs since the system contains eight sensors. It is believed that the inputs consist of the sensors' distance data.

Table 1. The sample of inputs and targets

SAMPLE OF 256 INPUT POSSIBILITIES																														
LF	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
RF	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
LR	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
RR	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
LSF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LSR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RSF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RSR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SAMPLE OF DESIRED TARGETS																														
S/B	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
SB	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	1	0	0
TL	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TR	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
GO	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X

By calculating the detection distance to the defined range, these data are normalized to be biner data. "1" is assigned to the distance data that fall within the range, and "0" is applied to the data outside of it. To put it another way, the value "1" is assigned when an item is detected, and the value "0" is assigned when none is. After normalization method, the probabilities of eight inputs are 2^8 or equal to 256. All the input possibilities are paired to desired targets and each input has a "one hot encoding" target, as it is seen in *table 1*.

The offline training is accomplished through the Arduino Due with an IDE program embedded in it. This method will determine the optimum weight values of the neural network.

The neural network is designed by referring to the basic of neural network formula:

$$y_j = \sigma\left(\sum_{i=1}^{n} w_{ij} x_i + b_j\right) \quad (1)$$

And the sigmoid activation function used in this system is using mathematics formula:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (2)$$

Backpropagation refers to the process of updating weights by changing old weights to new weights using a mathematical formula. The variation is determined by the difference between the output and target error deviations.

$$w_{new} = w_{old} - \alpha \frac{\partial E}{\partial w} \quad (3)$$

It is also done to value of bias,

$$b_{new} = b_{old} - \alpha \frac{\partial E}{\partial w} \quad (4)$$

After updating the weight and bias values, the signals are sent back to the network and repeating the process that has been done previously. This process will be repeated continuously until the optimum weight value is reached when the minimum error value is obtained, or after reaching a predetermined number of iterations. This iteration will be monitored by the mean square error (MSE) block to determine the error value.

To find the error, the formula used is:

$$E = \frac{1}{n}\sum_{1}^{n}(target_i - output_i)^2 \quad (5)$$

The training with the Arduino Due processor uses the Arduino IDE program with a C++ base including *math.h* library. Without any sensors connected, the program is embedded in the Arduino Due microcontroller. The Arduino Due is used to run the software, and the IDE serial monitor displays the results on the PC screen. To display the serial monitor, the Arduino needs to be connected to a PC via a USB connection port.

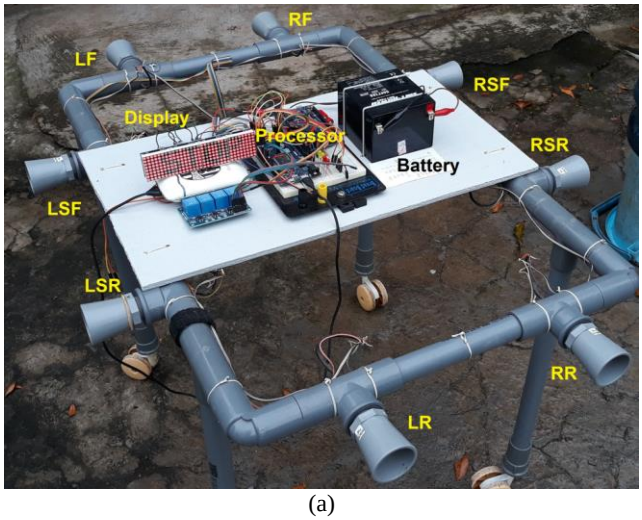
The neural network settings, including the number of patterns, input nodes, hidden nodes, output nodes, learning rate, etc., are first configured by the application. By specifying every value in the Arduino IDE program, every set of input and target is also configured in this network. A total of 256 patterns were selected for this investigation based on variations of 8 inputs. With a learning rate of 0.5, the neural network system was constructed with 16 hidden nodes and 5 output nodes. The learning success rate is 0.001 of the output versus target deviation, and the momentum value is 0.9.

Setting the initial weight value is the first stage in this training. This value can be adjusted at random and should ideally fall between -2 and 2. Prior to the hidden layer's neurons processing the generated weights, all the input data from the input array is multiplied by them. The product of this multiplication will be activated by the sigmoid activation function prior to being supplied to the hidden layer. The number of neurons in each layer determines the process that happens in the input to the hidden layers and in the hidden layer to the output layer.

Now that the neural network's final output has been produced, it is time to assess the system. If targets and outputs diverge, the network is back-propagated to examine and update all connection weight values. Back-propagation algorithm is used to achieve this procedure. The prior network is reached again by the back-propagation process. By using the momentum and learning rate values specified in the network parameter, it changes the connection weight value. Until the optimal values with the least number of errors between the outputs and the targets are determined, the procedure is repeated step by step and loop by loop. The software is processing all inputs and weights quickly until the ideal weight value is obtained because it is operating under the loop command. In the online training, as testing and implementation, the final optimal value will serve as a fixed-weight value.

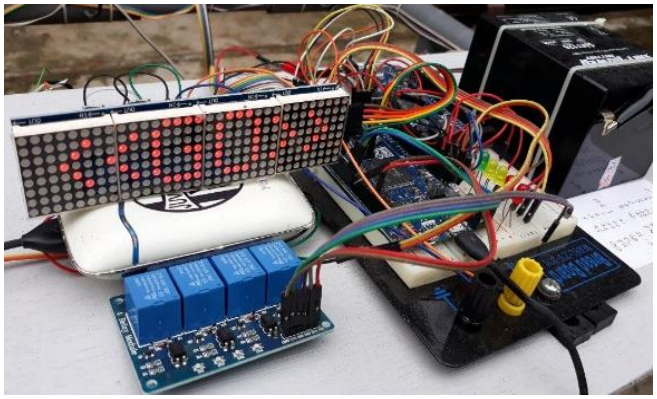
2.6. The Implementation Test

The Arduino Due is utilized to conduct online training connected to output devices and sensors, as seen in *figure 5*. The Arduino software used for this online training is the same as the Arduino software used for offline training by adding some libraries like MD_Parola.h, MD_MAX72XX.h, and SPI.h for action display, with the exception that the input and target data have been eliminated because the inputs will come directly from the sensors, and the output display has been used in place of the target to show whether the system is operating correctly.



(a)

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X



(b)

Figure 5. The hardware design of implementation, (a) All sides of sensors, (b) The processor and the dot matrix display

The system is no longer configured to determine the optimal weight values because here the connection weight values that came from the offline training are used.

Now, the values for the hidden weights and output weights are set as arrays. The data input from the sensors will be processed and multiplied for these arrays. Therefore, the purpose of the looping process is to immediately process the sensor data in the network until the precise required output is found, rather than updating the weight.

Table 2. The redefining actions of the outputs

Sensor	State	Range	Action	Degree	Display
LF	Near	50 – 300	Wide Turning Right	45	Right1
LF	Medium	301 – 600	MediumTurning Right	35	Right2
LF	Far	601 – 900	Narrow Turning Right	25	Right3
RF	Near	50 – 300	Wide Turning Left	-45	Left1
RF	Medium	301 – 600	Medium Turning Left	-35	Left2
RF	Far	601 – 900	Narrow Turning Left	-25	Left3

The outputs of turning left and right will be redefined to six actions (narrow turning left, medium turning left, wide turning left, narrow turning right, medium turning right, and broad turning right) after the desired outputs have been identified. The three states of the redefined actions (near, medium, and far) are determined by the distance of the detected item in front of the vehicle. This indicates that there are now nine complete actions, which include stopping, applying gentle brakes, and continuing straight. The redefining activities may be shown in *table 2*.

3. RESULTS

The offline training and implementation test were completed successfully, and the training process results are described here in two sections because of the process.

3.1. Training Results

The results of the training process are shown in the serial monitor window once the Arduino Due is connected to a laptop running the Arduino IDE software. The input and target data are provided, together with the generated output results, on the serial monitor for the first round of training. The network's error in this first training is very large, and the weight value is not suitable for online training. All training patterns including the outputs from the second and later training rounds are then shown on the serial monitor. The updated weight produced by the system shown in *tables 3 and 4* is displayed in the final round. The message shows the program's ideal weight numbers, indicating that the training problem has been resolved. This training is in 1000 iterations.

As shown in *figure 6*, the error begins to close to zero in the 300th iteration. And, after 1000 iterations, the output and target values are nearly identical, with an error value of 0.00198.

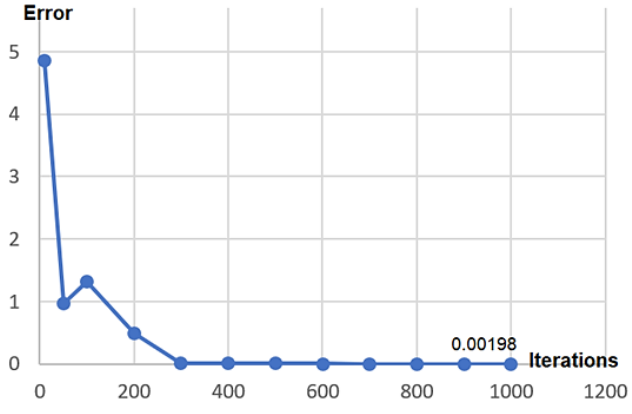


Figure 6. The MSE of training

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X

Table 3. The hidden weight values of Arduino offline training

Wx	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	-2.56581	-8.72781	-4.30023	5.95328	-2.12791	0.06568	-1.8863	-3.37449	-2.33458	6.64893	2.13669	-0.05946	-6.36897	2.68997	-1.86664	-4.87251
2	-5.47793	-6.38483	1.69712	-4.38289	2.9942	-0.58013	-6.54489	1.29994	4.057	4.15558	-4.92712	0.92275	-6.69279	-2.8959	-3.01331	4.11253
3	0.88993	0.96645	0.14119	-6.95148	-5.79182	-0.38104	0.32507	0.23752	-4.82198	3.95967	0.18154	-9.49399	0.32776	0.02189	0.35413	0.15652
4	0.86248	0.94721	0.11066	-7.00456	-6.04864	-0.27528	0.39951	0.26787	-4.99139	3.81453	0.16733	-8.77593	0.34348	0.04012	0.35434	0.1786
5	0.23973	-1.55135	-4.69508	1.23676	-4.88382	-0.39359	0.19715	-3.83344	0.69707	3.76416	-0.38792	1.4297	0.43569	-0.23437	-0.33641	-5.70011
6	0.29401	-1.44235	-4.51927	1.15342	-4.43389	-0.96406	0.10164	-3.67422	-1.8008	2.83217	-0.4315	1.80754	0.37052	-0.22115	-0.15642	-5.56779
7	-2.85607	0.03302	-0.39094	-5.74483	1.34774	-1.19026	-4.73218	-0.56365	-2.67648	-2.60622	-5.43777	1.57921	0.62285	-4.02216	-1.75356	-0.29958
8	-2.84077	0.10251	-0.33141	-5.96683	1.09894	-1.22855	-4.78416	-0.40895	-2.34897	-2.52327	-5.43028	1.93021	0.48835	-4.01735	-1.69326	-0.39519
b	2.7622	4.69936	0.54467	4.1877	3.33656	-1.7797	3.55675	0.51096	2.47929	-10.1777	0.22064	-0.95193	1.73642	-1.026	2.0769	-1.4557

Table 4. The output weight values of Arduino offline training

Wy	1	2	3	4	5
1	-3.84187	-4.05828	-1.54891	2.20115	2.07659
2	-6.10792	-7.0879	-0.8629	-4.67355	6.56953
3	-5.41033	-3.90107	4.19883	-1.55761	-0.33109
4	8.99452	-8.11549	-1.47903	1.1199	-0.43028
5	3.84823	-5.19075	0.41855	-1.04518	0.2573
6	-0.99468	-1.40155	-0.7727	-0.97696	-1.53473
7	-4.9484	-5.48333	-1.79049	3.89819	1.86448
8	-4.14014	-2.96622	3.32074	-0.75075	-0.70422
9	3.73155	-4.35358	0.22727	-0.36833	-0.82139
10	-7.46391	6.05179	-2.49853	-0.96876	-3.70752
11	-5.47625	-5.37293	-1.20653	5.19039	0.11657
12	7.02377	-8.55519	-1.30671	-0.71453	-0.37994
13	-4.4174	-6.1254	-3.53943	-4.57247	6.19837
14	-2.39324	-3.25451	-0.24253	4.37206	-2.41607
15	-2.42257	-3.18122	0.0477	0.13582	1.24078
16	-5.59847	-4.7908	7.77965	-1.68904	-2.58868
b	-5.14284	8.13051	-6.25358	-6.84615	-6.38369

The input and output of the system are tested by running it with dummy input and manually altering the random input before the program for online training implementation is executed. To find out how near the output is to the target, it is checked. *Table 5* presents the outcome of the training. The table demonstrates that, with a minimal error of 0.001, the output is reasonably near to the intended aim. To know and guarantee the detailed values generated by the outputs, the output value is intended to be five digits after the dot. Despite having identical goals, the output results could differ from one another. In the test and implementation of online training, the value greater than 0.9 will be set to 1, and the value less than 0.9 will be set to 0.

Table 5. The output of training

INPUTS	TARGETS	OUTPUTS				
0 0 0 0 0 0 0 0	0 0 0 0 1	0.00000	0.00000	0.00011	0.00045	0.99932
1 1 1 1 1 1 1 1	0 1 0 0 0	0.00000	1.00000	0.00016	0.00040	0.00004
1 0 0 0 0 0 0 0	0 0 0 1 0	0.00809	0.00000	0.00001	0.99660	0.00272
0 1 0 0 0 0 0 0	0 0 1 0 0	0.00229	0.00000	0.99862	0.00001	0.00010
0 1 0 0 1 0 0 0	1 0 0 0 0	0.99997	0.00000	0.00065	0.00035	0.00057
1 0 0 0 0 0 0 1	1 0 0 0 0	0.99998	0.00000	0.00022	0.00141	0.00139
0 0 1 1 1 1 1 1	0 0 0 0 1	0.00000	0.00479	0.00001	0.00000	0.99783
0 1 0 1 0 1 0 1	0 1 0 0 0	0.00040	0.99804	0.00329	0.00036	0.00214
1 0 1 0 1 0 1 0	0 1 0 0 0	0.00001	0.99987	0.00010	0.00156	0.00016
0 0 1 1 1 1 0 0	0 0 0 0 1	0.00000	0.00000	0.00000	0.00057	0.99900

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X

3.2. Implementation Test Results

As the primary input of the system, the ultrasonic sensors can be operated to such an extent that the circuit for the ultrasonic sensors can identify objects and determine their position and range in several tests. The processor is also capable of deciphering sensor signals and properly normalizing them in accordance with the specified range margin. The Arduino IDE serial monitor display, which displays the outputs based on the inputs from the sensor signal.

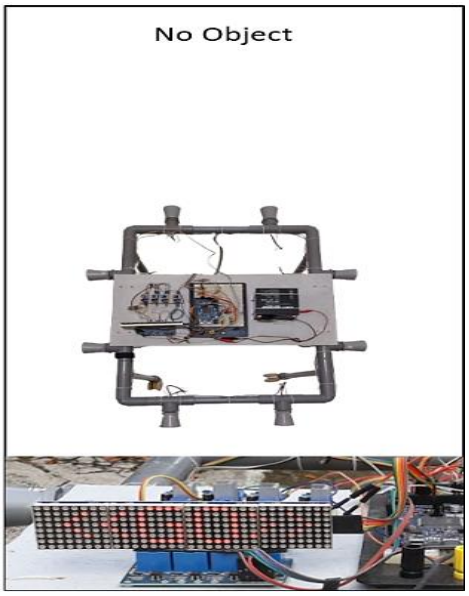
The time stamps show when the sensors began their detection procedure. The first five values represent the outputs: stop, soft braking, left, right, and straight. The inputs are the following eight values, which are separated by commas: LF, RF, LR, RR, LSF, LSR, RSF, and RSR. Table 5 shows that each line contains five outputs, at least one of which has a value of 0.9. This value denotes the necessary course of action. Six actions are included in the turning action set, with a tree action for each turn, whether it be left or right, depending on the object's determined distance.

The system is ready to receive sensor input and process it straight to the output after all weight values have been implemented into the system network. A led matrix display is used to design the output, and each output is assigned a word based on the action that the system is supposed to perform.

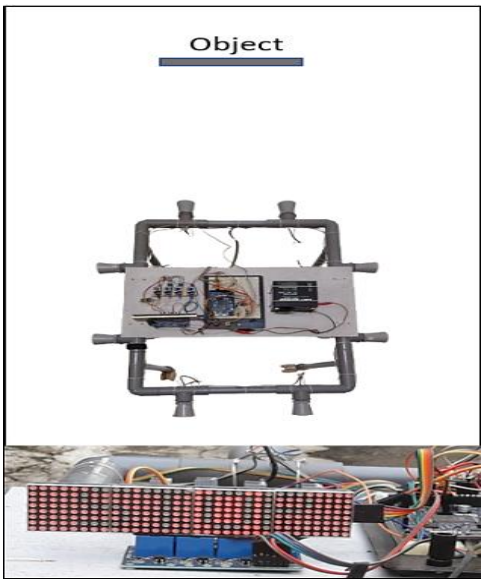
The word "STOP" denotes the beginning of the braking process by the system to bring the car to a complete stop. The acronym "SOFT" stands for "soft braking," which refers to the use of gradual braking by the car's system to prevent rear-end collisions with other vehicles. The terms "LEFT1" and "LEFT2" denote turning left at approximately -25 degrees, -35 degrees, and -45 degrees, respectively. "RIGHT1" indicates a turn to the right of roughly 25 degrees, "RIGHT2" a turn to 35 degrees, and "LEFT3" a turn to 45 degrees. "GO" is the final command, which the car continues to move forward. The displays of the events that occur when an object is positioned in front of the sensors are shown in *figure 7*.

Table 6. DET: Object Detection, ACT: Action Display

INPUTS								OUTPUTS					ACTIONS	
LF	RF	LR	RR	LSF	LSR	RSF	RSR	ST	SB	TL	TR	GO	DET	ACT
898	897	541	313	456	453	200	82	0.000	0.000	0.003	0.000	0.996	NONE	GO
667	705	679	688	484	499	509	483	0.990	0.005	0.003	0.001	0.002	ALL	STOP
200	904	536	310	450	451	190	193	0.992	0.000	0.001	0.000	0.000	ALL	STOP
672	535	354	342	471	496	498	497	0.005	1.000	0.000	0.002	0.000	ALL	SOFT
931	481	331	325	198	196	396	383	0.003	1.000	0.000	0.002	0.000	ALL	SOFT
909	702	511	484	511	509	489	476	0.000	0.002	0.989	0.003	0.002	FAR	LEFT1
831	351	544	317	450	451	198	198	0.015	0.000	0.984	0.000	0.002	MED	LEFT2
888	198	497	520	502	494	517	495	0.007	0.000	0.987	0.003	0.005	NEAR	LEFT3
676	923	356	510	642	544	212	210	0.002	0.000	0.000	0.985	0.000	FAR	RIGHT1
516	910	510	510	490	485	481	511	0.005	0.002	0.000	0.988	0.001	MED	RIGHT2
207	914	492	507	486	516	512	484	0.005	0.003	0.002	0.988	0.000	NEAR	RIGHT3

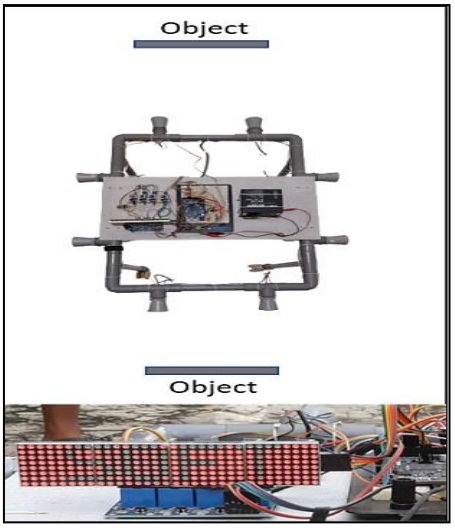


(a)

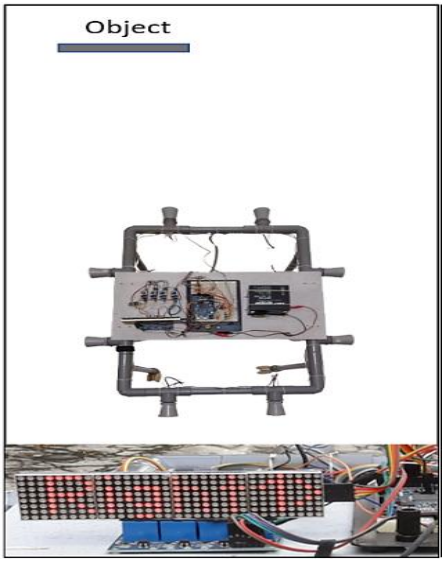


(b)

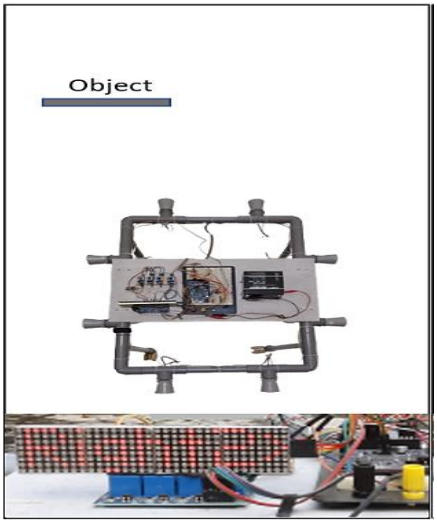
FOREX Publication
Open Access | Rapid and quality publishing
International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X



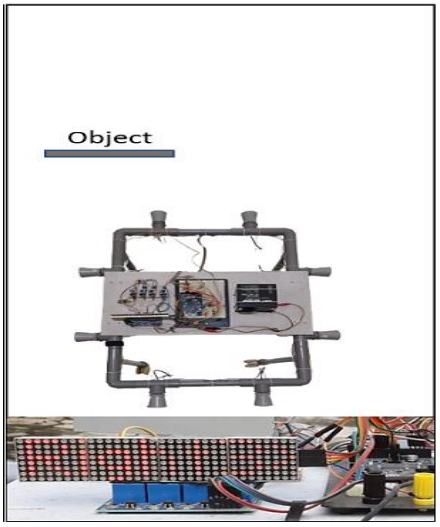
(c)



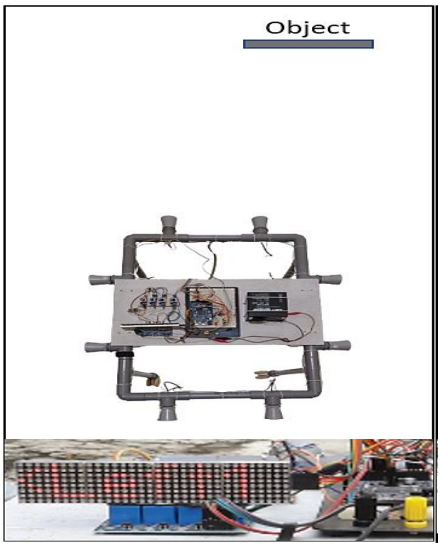
(d)



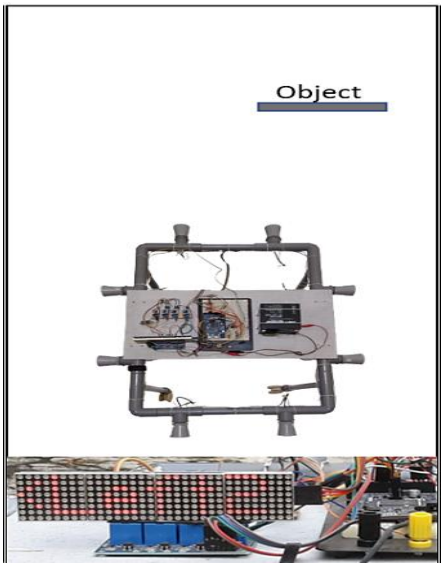
(e)



(f)

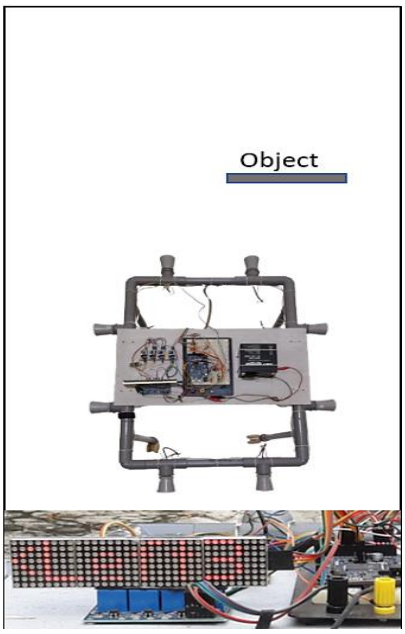


(g)



(h)

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 3 | Pages 870-880 | e-ISSN: 2347-470X



(i)

Figure 7. The display of the actions, (a) Going ahead action, (b) Soft braking action, (c) Stop or braking action, (d) Wide turning left action, (e) Medium turning left action, (f) Narrow turning left action, (g) Wide turning right action, (h) Medium turning right action, and (i) Narrow turning right action.

Table 5 displays the possible sample of sensor detection based on object distance. If the system detects nothing in front of the car, it will press "GO." If an item is detected by one or both front sensors and one or more of the opposite side sensors as well, "STOP" will be triggered. The "STOP" operation is carried out by assuming that the back sensors have not picked up any objects. The system will take "SOFT" action if any items are detected. Depending on the position and distance of the object, turning left or right will be executed if one of the front sensors detects an object and the opposite side sensors do not detect any. The distant object turns narrowly by using the "LEFT1" or "RIGHT1" commands. A medium-range object will trigger a "LEFT2" or "RIGHT2" action from the system. Additionally, a close item will cause the system to turn widely by using the "LEFT3" or "RIGHT3" commands. As seen in *figure 7*, the action words are presented in a dot-matrix display. The results of implementation test are proof that the system runs well according to the desired target.

4. DISCUSSION

It is evident from going through several ANN offline training and implementation procedures that the weight values of each network are what give the neural network approach its strength. Given the length of the training process, it is advised to complete offline training before obtaining the network value rather than doing it instantly online. Therefore, although a high-speed processor handles it, it is not able to directly use it to conduct the actions required by the system. Additionally, when it's used on cars to keep things from getting in the way.

When conducting the offline training using the Arduino Due, the error value obtained at the 300th iteration is 0.00859, which means that the output and target are equal in value. Due to its limited memory and processor speed, the Arduino iteration process is very slow, in about 14 minutes for 1000 iterations. But in implementation test, Arduino did well as expected. Therefore, it is recommended to use high speed processor with sufficient memory to run offline training. And also it is better to use same type of processor and same ANN algorithm in offline training and implementation.

To ensure that the online training is successful, both offline training approaches can generate network weight values that can be used in the online training. It indicates that the Advanced Artificial Neural Network (AANN) approach was applied to this investigation with success. Moreover, this approach can be extended to more comprehensive applications in subsequent research.

To improve the system, long range ultrasonic sensor with continue signal can be applied, that the sensors are independent to the refresh rate and more real time data can be obtained. The fuzzy base control also can be combined to the system to add performance in detecting the object by defining detection range categories in member set function.

Future work of this study is being implemented on autonomous vehicle unit and being collaborated with another sensor system in the vehicle, so it can improve object detection around the vehicle. And the potential application of this method can be implemented to all types of land vehicle, especially for vehicle that has no own track and needs more safety system.

5. CONCLUSIONS

When integrated into a multi-sensor circuit, the ultrasonic long-range sensor can be utilized as an electric vehicle sensor to identify obstacles and prevent crashes without depending on illumination, climate, or hue. The system that incorporates the ANN approach, both offline training and implementation test processes these sensors successfully. The goal of the offline training is to identify the neural network's ideal weight values with the least amount of error. The online training is carried out to apply the ideal connection weight values identified by the offline training to the vehicle's neural network system. With an error of 0.001 per 1000 iterations, the Arduino Due is used for the offline training. The outputs of the algorithm are in accordance with the intended goals, and the system is functioning flawlessly overall. The redefinition of actions by considering the object's distance is superior to using a neural network alone since the system can sense the vehicle's surroundings with greater detail and avoid obstacles with precision. In subsequent research, the advanced ANN approach will be used in an actual autonomous car in conjunction with additional navigation sensors and lateral driving techniques to achieve the desired output actions.

Author Contributions: Conceptualization, E.N.B and S.A.D.; methodology, E.N.B.; software, E.N.B and B.A; validation, S.A.D., A.A.N and M.H.; formal analysis, M.K.; investigation,

E.N.B.; resources, E.N.B.; data curation, E.N.B.; writing—original draft preparation, E.N.B.; writing—review and editing, E.N.B. and M.K; visualization, E.N.B.; supervision, S.A.D and M.H.; project administration, A.A.N.; funding acquisition, A.A.N. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: Authors thank to Industrial Technology Faculty of Universitas Islam Sultan Agung Semarang Indonesia for the supports.

Conflicts of Interest: Authors declare there is no conflict of interest in this manuscript.

REFERENCES

[1] N. Ding, K. Prasad, and T. T. Lie, "The Electric Vehicle: A Review," vol. 9, no. 1, pp. 49–66, 2017.

[2] WHO, "Road Traffic Injuries," https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries. [Accessed: 14-Feb-2022].

[3] K. Saleh, M. Hossny, and S. Nahavandi, "Effective Vehicle-based Kangaroo Detection for Collision Warning Systems using Region-based Convolutional Networks," Sensors (Switzerland), vol. 18, no. 6, 2018.

[4] S. Bayles H and S. Davidson, "Driverless Cars and Accessibility," Washington, DC, 2019.

[5] F. Zhang, C. M. Martinez, D. Clarke, D. Cao, and A. Knoll, "Neural Network Based Uncertainty Prediction for Autonomous Vehicle Application," Front. Neurorobot., vol. 13, no. May, pp. 1–17, 2019.

[6] S. O. Bamgbose, X. Li, and L. Qian, "Trajectory Tracking Control Optimization with Neural Network for Autonomous Vehicles," Adv. Sci. Technol. Eng. Syst., vol. 4, no. 1, pp. 217–224, 2019.

[7] M. C. De Simone, Z. B. Rivera, and D. Guida, "Obstacle Avoidance System for Unmanned Ground Vehicles by Using Ultrasonic Sensors," Machines, vol. 6, no. 2, 2018.

[8] S. Sugathan, B. V. Sowmya Shree, M. R. Warrier, and C. M. Vidhyapathi, "Collision Avoidance using Neural Networks," IOP Conf. Ser. Mater. Sci. Eng., vol. 263, no. 5, 2017.

[9] T. A. Andreeva and W. W. Durgin, "Wind Tunnel Investigation of Sound Attenuation in Turbulent Flow," Ultrasonics, vol. 61, pp. 15–19, 2015.

[10] T. F. Wu, P. S. Tsai, N. T. Hu, and J. Y. Chen, "Use of Ultrasonic Sensors to Enable Wheeled Mobile Robots to Avoid Obstacles," Proc. - 2014 10th Int. Conf. Intell. Inf. Hiding Multimed. Signal Process. IIH-MSP 2014, pp. 958–961, 2014.

[11] D. Jeon and H. Choi, "Multi-sensor Fusion for Vehicle Localization in Real Environment," ICCAS 2015 - 2015 15th Int. Conf. Control. Autom. Syst. Proc., no. Iccas, pp. 411–415, 2015.

[12] L. Alonso et al., "Genetically Tuned Controller of an Adaptive Cruise Control for Urban Traffic Based on Ultrasounds," pp. 479–485, 2010.

[13] E. N. Budisusila et al., "Real Time System Handling Using Multi Fixed Weight Artificial Neural Network," Int. Conf. Electr. Eng. Comput. Sci. Informatics, no. September, pp. 504–508, 2023.

[14] W. Bruce and E. V. O. N. Otter, "Artificial Neural Network Autonomous Vehicle Artificial Neural Network Controlled Vehicle," 2016.

[15] T. P. Le, L. V. Dat, and I. Stiharu, "Application of Neural Networks to Design The Controller for Autonomous Vehicles by Learning Driver's Behavior," 2013 Int. Conf. Control. Autom. Inf. Sci. ICCAIS 2013, pp. 1–6, 2013.

[16] C. Tîrnăucă, J. L. Montaña, and Z. Mediavilla, "Neural Networks in Autonomous Driving," AAAI Spring Symp. - Tech. Rep., vol. SS-17-01-, pp. 520–523, 2017.

[17] E. N. Budisusila, S. Arttini, D. Prasetyowati, and B. Y. Suprapto, "Neural network training for serial multisensor of autonomous vehicle system," vol. 12, no. 5, pp. 5415–5426, 2022.

[18] G. Ognjanovski, "Everything you need to know about Neural Networks and Backpropagation," Hacker Noon, pp. 1–5, 2019.

[19] A. Budianto et al., "Analysis of Artificial Intelligence Application using Back Propagation Neural Network and Fuzzy Logic Controller on Wall-following Autonomous Mobile Robot," 2017 Int. Symp. Electron. Smart Devices, ISESD 2017, vol. 2018-Janua, no. 1, pp. 62–66, 2017.

[20] P. Baldi, P. Sadowski, and Z. Lu, "Learning in The Machine: Random Backpropagation and The Deep Learning Channel," Artif. Intell., vol. 260, no. 1, pp. 1–35, 2018.

[21] B. Arifin, B. Y. Suprapto, S. Arttini, D. Prasetyowati, and Z. Nawawi, "Steering Control in Electric Power Steering Autonomous Vehicle Using Type-2 Fuzzy Logic Control and PI Control," Word Electr. Veh. J., vol. 13, no. 53, 2022.

[22] Y. Pan, C.-A. Cheng, K. Saigol, and K. Lee, "Learning Deep Neural Network Control Policies for Agile Off-Road Autonomous Driving," no. Nips, 2017.

[23] H. J. Vishnukumar, "Machine Learning and Deep Neural Network – Artificial Intelligence Core for Lab and Real-World Test and Validation for ADAS and Autonomous Vehicles," Intell. Syst. Conf. London, UK, no. September, pp. 714–721, 2017.

[24] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," Int. J. Comput. Appl., vol. 175, no. 4, pp. 7–9, 2017.

[25] Maxbotix, "HRXL-MaxSonar ® - WR TM Series," 2012.