PHILIPPINE ENGINEERING JOURNAL
PEJ 2021; Vol. 42, No. 2: 57-74

Received: 20 May 2021
Accepted: 09 November 2021

Comparative Analysis of Fuzzy-Neural Network Implementations on an Autonomous Electric Vehicle

John Paolo A. Ramoso[1*], Manuel C. Ramos, Jr.[2]
1 Department of Electrical Engineering, University of the Philippines Los Baños
2 Electrical and Electronics Engineering Institute, University of the Philippines Diliman
**Corresponding Author: jaramoso@up.edu.ph*

Abstract— *The aim of this research is to implement an optimized hybrid fuzzy-neural (FN) algorithm for an autonomous electric vehicle's stop-and-go decision-making and control. Four (4) different algorithms (purely fuzzy logic (FL), one (1) hidden layer (H1) FN, two (2) hidden layers (H2) FN, and purely neural network (NN)) were deployed in a buggy-type electric vehicle (EV) to compare their performances in real road conditions. The test EV was equipped with a LiDAR Lite sensor which served as the range finder to measure headway distance while an optical flow sensor and the motor's built-in hall sensors were used to measure speed. The EV was also retrofitted with a dsPIC30F4011 microcontroller for processing and control. Both indoor and outdoor road tests were conducted to compare the difference between a controlled environment (well-lit with good road conditions) versus actual road conditions (including physical limitations), respectively. It was observed in the indoor tests that increasing the hidden layers from H1 to H2 made the algorithm more robust and decreased jerking phenomenon when the vehicle was stationary. Results from the outdoor tests also revealed that FN network with H2 (successful in eight (8) out of ten (10) runs) had better control in maintaining proper headway distance and more fluid transition in acceleration and deceleration. Hardware considerations were also outlined focusing on deploying machine learning codes and weights to a microcontroller. The ~56kB initial code size was way above the allowable 48kB program memory of the microcontroller therefore the data type of the weights were changed to shrink the code to ~38kB.*

Keywords— Artificial Neural Networks, Autonomous Vehicle, Fuzzy Logic, Fuzzy-Neural Network

I. INTRODUCTION

The substantial increase in traffic congestion in the country has been a major concern due to its negative impacts such as delays, unnecessary length of time on the road, high fuel consumption, wear and tear of vehicles, and as well as stress and frustration for motorists which often lead to road rage.

Autonomous vehicular control systems and traffic management algorithms can ease up traffic in any roadway. Numerous proposals in traffic management, both macroscopic and microscopic [1], have been presented in other papers in recent years but were usually done in simulation. These algorithms must be translated to physical verifications and control system analyses [2] to account for other phenomena that cannot be observed in simulations.

Early models developed in vehicular modeling were the Stimuli-Response [3], Action Point and Safety-distance [4], and optimal velocity [5]. These microscopic approaches describe vehicular movements and decision-making . These methods focus on the stop-and-go movements of the vehicle in control. As technology advancements became more available to researchers, macroscopic approaches to traffic management emerged. Cellular Automata [6] and Trajectory-based [7] models were among the early models describing vehicular interaction in a third-person perspective resulting in increased efficiency of acceleration. These models do not depend on a single vehicle's parameters but also factor in all the other vehicle's parameters in the given system.

Many advancements in autonomous electric vehicle's (EV's) stop-and-go decision-making models could be seen in recent literature [8] [9]. Basic neural networks were usually used in specific roles like in the prediction of vehicle conflicts [10] [11] or actively controlling specific mechanical parts of vehicles [12] [13]. There were also advancements in research in electric vehicles being autonomous but connected [14] [15] at the same time. These different models can be attributed to various machine learning techniques and method variations and adoptions consequently applied to distinct roles or applications. However, only a few studies reached the implementation stage and used real-world data for their training and validation tests.

A fuzzy logic (FL) inference system has been applied to various control processes such as powertrain controllers to optimize fuel consumption [16] or using a manipulator control for navigation [17]. Likewise, artificial neural networks (NNs) were applied to autonomous driving or vehicles for safe lane-changing models [18] or pedestrian-vehicle conflict predictions [19]. Literature also established a lot of alterations and a mix of fuzzy-neural (FN) networks that worked with a four-wheel-steering vehicle [20], a variety of unmanned aerial, underwater, and floating drones, and even hypersonic vehicles [21]. Even intelligent cruise control in highways can be implemented with FN controllers [22]. It used a base structure of an FL inference system to either mimic an expert system or an arbitrary control program wherein the defuzzification part was replaced with a neural network hidden layer or layers. Thus, this enhanced the strength of both algorithms; FL with machine interpretation of independent variables and NN with processing data to recognize trends [23].

An FN algorithm was then proposed for development and optimization then compared its performance to both purely FL and NN algorithms. Movements from EVs driven by experienced e-trike drivers were gathered and used as the expert system training dataset for algorithm optimization. An EV buggy fabricated at the Electrical and Electronics Engineering Institute (EEEI) of the University of the Philippines Diliman (UPD) was retrofitted with fabricated sensor modules and microcontroller board. Testing was confined inside the university's premises for road safety.

II. METHODOLOGY

2.1 Optimized Algorithm and Simulated Driving

Four (4) algorithms namely: an FL controller, two derivations of an FN algorithm – a single hidden layer (H1) and two (2) hidden layer (H2) FN topologies, and a fully-connected NN were developed to test different EV performances and to observe the following vehicle's (FV's) algorithmic response to the lead vehicle (LV). The assumed target behavior was that the FV must maintain a certain distance from the LV. All algorithms were created to have two (2) inputs (relative velocities (RV) and headway clearance or distance divergence (DSSD – distance sum of squared difference)) and one output (Acceleration).

The FL inference system established had five (5) membership functions (MF) per input. The MF for the DSSD were much too close (MTC), too close (TC), okay (K), too far (TF), and much too far (MTF). On the other hand, the MF for RV were closing fast (CF), closing (CL), match (Z), opening (OP), and opening fast (OF). These MF were relatively observable behavior of the distance headway and relative velocity between the LV and FV. These MF were then fed to 25 rule-based inference nodes to map out the interaction of the different MFs of an input to the other input variables. A Dombi t-norm, known for its high classification accuracy and easier code implementation [24], was used on each inference to determine the minimum value of the antecedents. This inference determines the different scenarios the system responds to. After the inference part of the algorithm, the results from the rule-based inference nodes were then collected in a defuzzifying node. This node took into consideration the supposed different reaction of the system to the input it was presented with. The activation function used was a TSK (Takagi-Sugeno-Kang) function so that the acceleration was interpreted by crisp output values [25] [26]. The defuzzification outputs the response of the system in terms of acceleration in five (5) different singleton values: strong acceleration (SA), light acceleration (LA), constant acceleration (C), light deceleration (LD), and strong deceleration (SD). These output accelerations were the responses of the EV (the FV in the setup) to maintain a good headway from the LV. Table 1 shows the acceleration rule matrix of the FL system while Figure 1 shows the three-dimensional representation of MF values highlighting the acceleration values in the range of input MFs. The resulting FL system architecture is shown in Figure 2 to illustrate the connection between nodes and phases.

Table 1. Rule-based Inference used for the Fuzzy Logic controller

Acceleration		RV				
		CF	CL	Z	OP	OF
DSSD	MTF	C	LA	LA	SA	SA
	TF	C	C	LA	LA	SA
	K	LD	C	C	C	LA
	TC	SD	LD	LD	C	C
	MTC	SD	SD	LD	LD	C

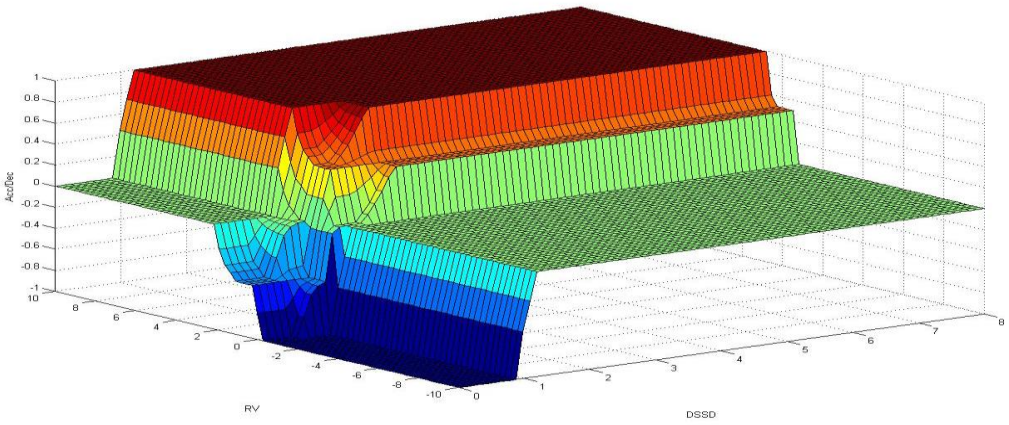

Figure 1. A three dimension graphical representation of the interaction between the inputs (DSSD and RV) to its output (acceleration)

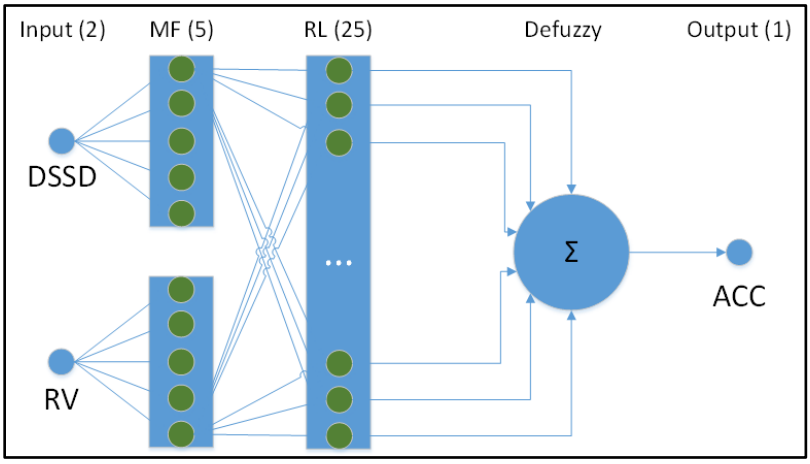

Figure 2. Fuzzy Logic controller that takes in DSSD and RV as inputs and outputs ACC showing the connection and process of the inference

An FN network was then developed from the FL system discussed above. The defuzzifying node was replaced with a hidden layer of NN. The idea was to replace the centralized defuzzifying function with smaller decentralized processes [8] [27]. To match the 25 rule-based inference, a single hidden layer of 25 nodes was attached in place of the defuzzifying node. Another hidden layer was added to increase learning capability, but this also increased the computational burden on the processing module. The second hidden layer was structured with 15 nodes that is fully connected to the first hidden layer. Figure 3 shows both the H1 and H2 FN architectures to highlight the connection of each node from layer to layer.

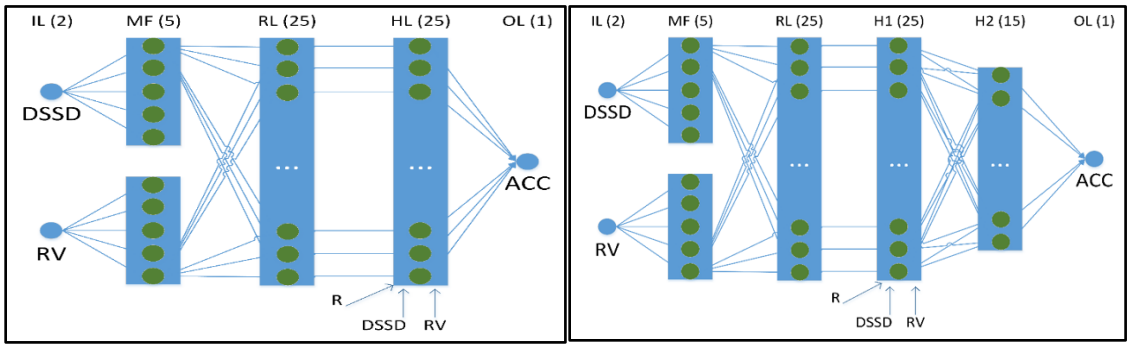

Figure 3. Proposed Fuzzy-Neural Networks with one hidden layer (left) and with two hidden layers (right) to be deployed to the EV

A fully connected neural network was then developed to serve as the baseline with the FL-only algorithm of the study as adapted from [11]. A two (2) hidden layer with five (5) nodes each was adapted to accept the RV and DSSD as inputs and acceleration as its output. The network is shown in Figure 4.

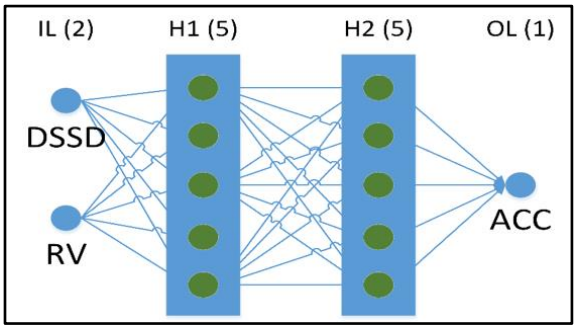

Figure 4. A fully connected Artificial Neural Network with two hidden layers used as a baseline parallel to the fuzzy logic only algorithm

The algorithms (FL, H1, H2, and NN) were then trained, validated, and tested with a total of 2744 data points gathered from seven (7) UPD e-trike drivers with a variety of EV driving experiences. The methodology and characterization of the data set was outlined in a previous publication [28]. Fifty percent (50%) of the data set was delegated for training, twenty-five percent (25 %) for validation, and the remaining twenty-five percent (25%) for the test set. This was arbitrarily set for ease of dividing the total data points. A sample of the data collected from two (2) e-trike drivers can be seen in Figure 5. Results show how driver three (3) was more defensive in driving and maintaining the e- trike's headway clearance while driver seven (7) was more aggressive as evident in the clustering of data points to stopping than maintaining velocity at a distance.

The algorithms' weights were then optimized at selected hyperparameters: starting random weights W1 (from negative one to positive one) and W2 (from negative two to positive two), learning rates Co (constant) and De (decaying), and batch processing Ba (full batch) and mB

(minibatch). These hyperparameters then multiplied all network algorithms (all algorithms except the purely fuzzy logic controller) into eight (8) variations each.

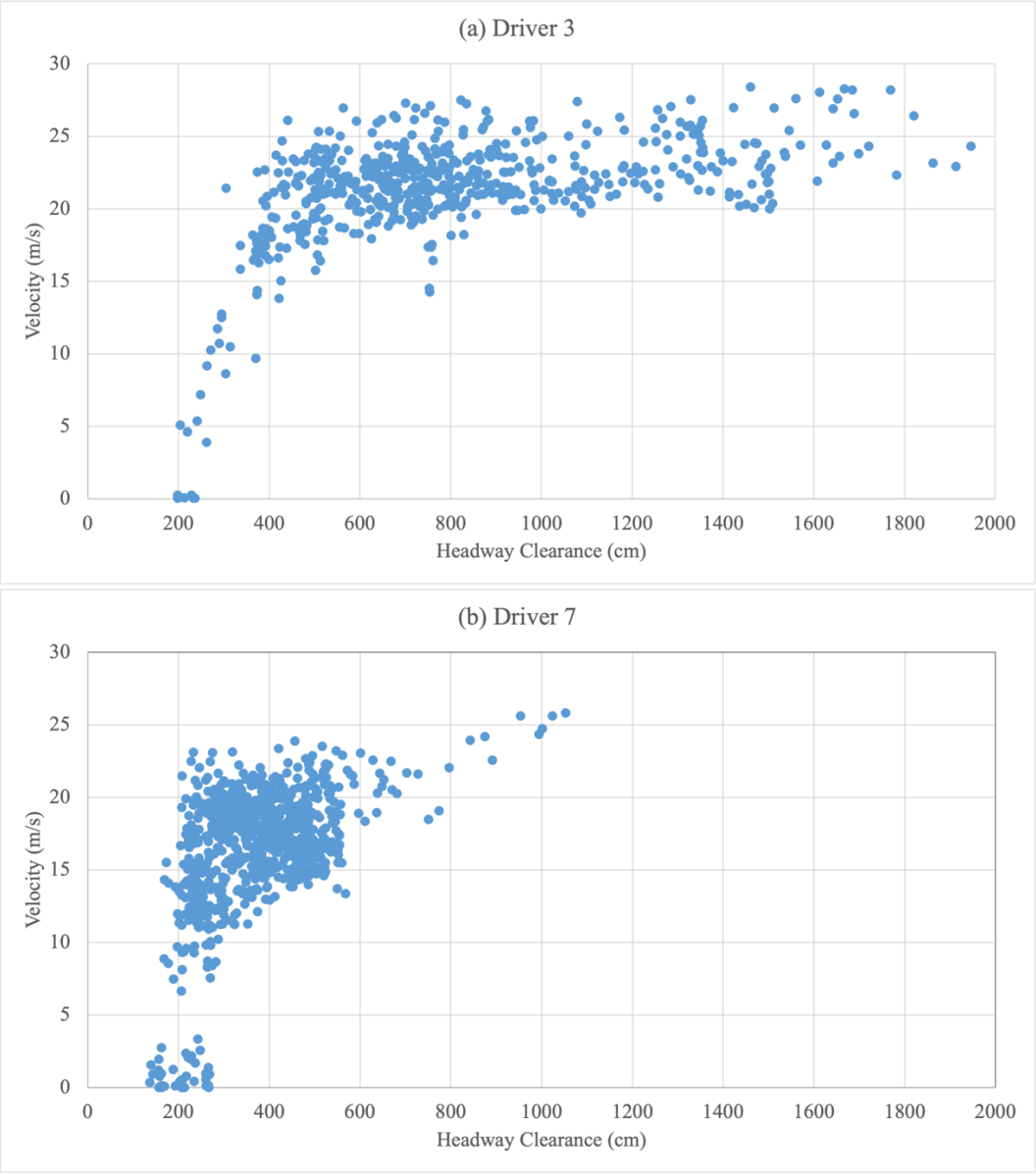

Figure 5. Sample data set from driver 3 and driver 7 showing difference in driving preference

Consequently, these algorithms were subjected to situational simulations and were then culled. Those variations of algorithms that were not able to maintain a head way (crashed) with the LV were removed from the list to be tested in hardware. Figure 6 shows a sample of failed and successful simulation runs. The selected optimized parameters after the simulations were W1CoBa, W1DeBa, W2DeBa, and W1ComB. These four (4) optimized parameters were used for the three (3) neural network topologies (NN, H1, and H2) and were benchmarked against

the FL-only algorithm. A total of 13 optimized algorithms were loaded to the fabricated EV buggy for hardware indoor testing and verification in actual road conditions.

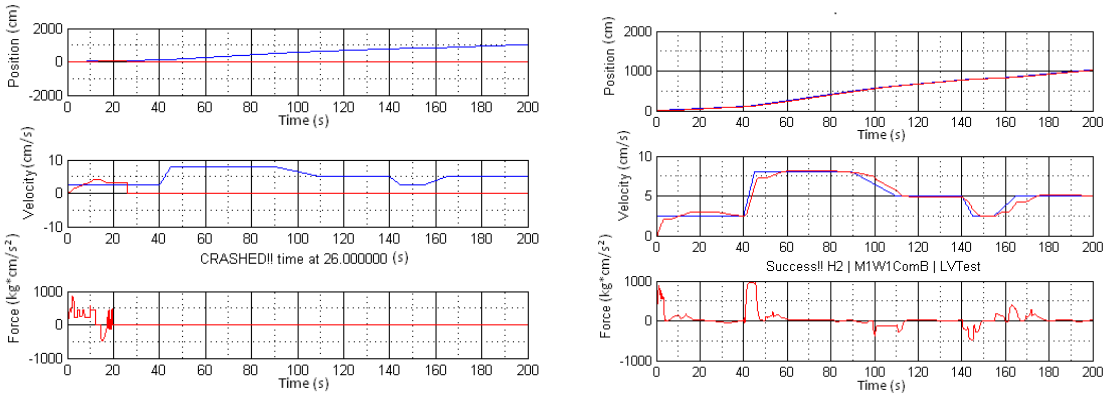

Figure 6. Sample simulation runs of a crashed FV failing to maintain headway from LV (left) and a successful simulation (right).

2.2 Hardware Design

An autonomous EV was built at the Robotics Automation Laboratory at EEEI, UPD. The EV was at its early stages and had the essential car parts only as seen in Figure 7. The EV buggy was 2.3m long by 1.2m wide by 1.4m tall. It was retrofitted with a microcontroller for the test algorithms and an array of sensors to collect input data.

Figure 7. Fabricated EV's frame with the essential car parts only: chassis, in-tire motors, battery packs, and controller.

A dsPIC30F4011 Microchip controller was used to control motor speed, gather data from the sensors, and house the test algorithms. It used a C compiler optimized instruction set architecture and dedicated 24-bit wide instructions and 16-bit wide data path [29]. The controller was selected due to multiple attributes. The EV's power supply will be dependent on a battery pack that provides power to the controller and motors. The selected controller has a wide operating voltage range (2.5V to 5.5V) which would accommodate the voltage swings due to the motor transient demands. The controller also offers a variety of timer modules and multiple duty cycle generators with a 10MHz oscillator. With the primary oscillator selected, an x16 multiplier (XT w/PLL 16x) was used to bring the clock speed to 140MHz. This was also important due to the multiple sensors data that the controller needs in order analyze the optimized test algorithms. Three (3) different timers (from two (2) independent 16bit timers) were also used to support different parts of the algorithm: the first one for the main code and the other two (2) for the different interrupt flags. These timers and flags were important to make sure the sensors could signal to the EV if it needs to stop for its safety. The microcontroller must also support multiple communication protocols (SPI, I2C, etc .) which must be flexible enough to adapt to a sensor's needs. The controller also offers a 10-bit analog-to-digital module, PWM-able pins, and sufficient digital and analog pins to support the designed hardware's framework. The hardware's block diagram can be seen in Figure 8. The figure shows the control and data flow of the system.

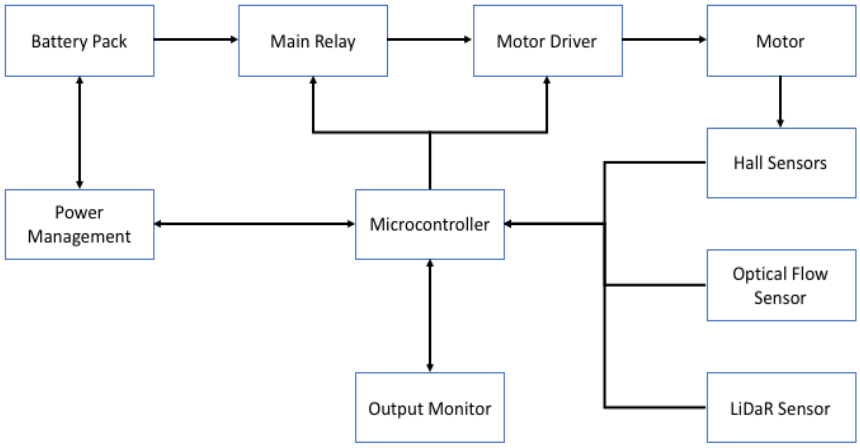

Figure 8. The hardware's block diagram showing the connection of the different parts of the system.

The hardware's framework was clustered into three (3) main parts: the sensors, the actuators, and the main board. The sensors were a LiDAR range finder, an optical flow sensor, and the built-in hall sensors in the tires. These sensors continuously fed data to the microcontroller. With timed intervals, the microcontroller selects which sensor to read and process.

The actuators were mainly the motor drivers which controls the wheels directly. There were two (2) 1kW 60V brushless DC (BLDC) motors with 1.32m wheel circumference that were equipped with hall effect sensors. The motors were controlled by varying the resistance in

parallel with the input terminals. This was done using X9C103 digital potentiometers operated by the microcontroller. The main switch for the motor drivers was a KZJ60VDC main relay contactor. The other central components of the framework were the microcontroller and power management block. A battery array of three (3) 100Ah and two (2) 60Ah Sinobatt batteries were loaded into the EV buggy. An output monitor was added to serve as a realtime data display.

Input data, DSSD and RV, were derived by measuring the headway between the FV and LV and determining the FV's velocity with respect to the LV's velocity, respectively. For the Headway, the LiDAR-lite v1 was used to determine the distance between the LV and FV. LiDAR emits a laser and clocks how long it takes the laser to get back to the LiDAR. The measured time was then multiplied with the speed of the laser. All these computations were done inside the LiDAR and the acquired data from the LiDAR to the microcontroller were set as raw measured distance. It operated at 5V nominal input voltage which was the same as the hardware system developed. The datasheet specified that it has a maximum range of 40m under typical conditions. However, field testing indicated that it can only accurately measure up to 30-35m. This was placed at 0.2m indented in front of the test EV. This ensured that the lidar was over the ~0.2m minimum distance requirement. Some errors in reading the LiDAR were observed during the fabrication and trial phases. Troubleshooting of the LiDAR was mostly focused on the communication side, mounting design, and laser feedback. The communication between the microcontroller and the LiDAR should be initially calibrated correctly to match clock timing. The mounting of the LiDAR on the EV was originally at the angle of horizon (at 0 degrees) and 0.3m above the ground.

A redundancy setup of an optical flow sensor and the built-in hall sensors of the motors were used for the FV's velocity determination. The ADNS3080 Optical Flow Sensor was calibrated with the height of 0.2m from the ground (it was placed higher than recommended to protect the sensors from the road conditions of the tests). The electric wheels used on the EV has built-in hall sensors that return pulses as magnetic poles in the tires which passes through the sensor. It was observed that the optical flow sensor's data reliability was dependent on the lighting condition and the texture of the road. The hall sensors' data were observed to be unreliable on abrupt changes in acceleration. The hall sensors have slow reaction outputs to the sudden changes in acceleration thus the data seemed to be unreliable. The two (2) sensors were used to complement each other's data due to inherent flaws in each other's means of measuring. A simple algorithm was developed to achieve this. It was observed that most of the time, the two (2) sensors gave the same data. But the abrupt changes in velocity and acceleration tend to give hall sensors a momentary false data. While certain areas in the Outdoor Test rendered the optical flow sensor a null output except for slow velocities. The redundancy of the two sensors gave the system more precise measurements of the data needed for the algorithm's optimizations.

2.3 Conditions for road testing

The EV loaded with the optimized algorithms were then tested to react or follow an LV in an indoor and an outdoor track. The four (4) algorithms calibrated with the four (4) optimized hyperparameters (except for the FL-only algorithm) were uploaded into the microcontroller for indoor road testing.

The indoor test was designed to simulate good road conditions with 10m road length. The road track was well lit, relatively clean of dust and dirt, and leveled. The sufficient lighting of the track was for the optical flow sensor. Road markers were placed on the tracks as reference frames to verify velocity and acceleration observations as seen in Figure 9. An obstacle was placed at the 8m mark to simulate enough clearance for acceleration then followed by an expected deceleration and eventual braking. All tests were done with five (5) trials each with the acceleration of the EV recorded (the minimum and maximum accelerations were noted). The indoor test was implemented in the High Voltage Laboratory at the EEEI.

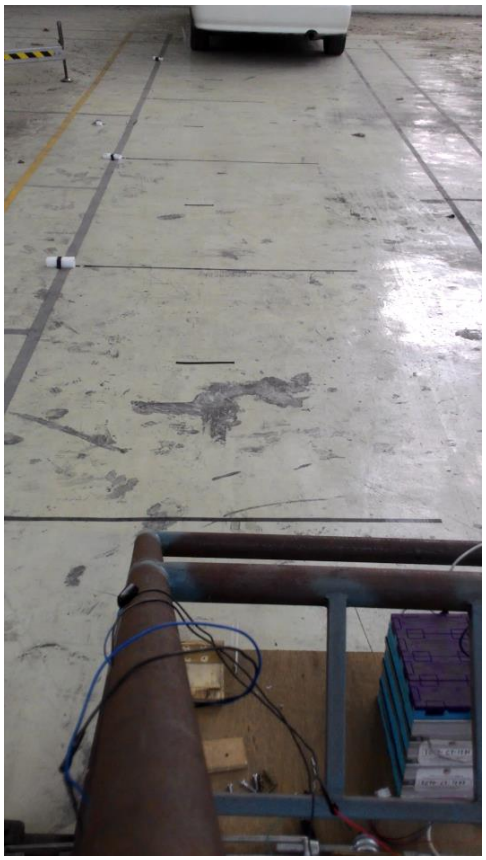

Figure 9. A snippet from the recordings of an indoor test showing the road markings in the indoor test

Results from the indoor tests determined the performance and appropriate optimized hyperparameters used in the outdoor tests. The algorithm variation with the greatest number of successes (stopped at appropriate stopping distance) and the best driving experience (realistic acceleration and deceleration profile) was chosen to move on to the next testing phase.

The outdoor testing, on the other hand, was done in the parking lot and around the EEEI building. The indoor test's setup was replicated (10m track with an obstacle at the 8m mark). The test was designed to imitate real road conditions with unpredictable lighting intensities and imperfect and unleveled paved cement conditions. The variability of the sensors' input values

due to real-world factors were critical in determining the robustness of the algorithms and its optimized parameters. All neural network topology outdoor tests were done with eight (8) trials each while the FL topology with only four (4) trials.

III. RESULTS AND ANALYSIS

3.1 Indoor Tests

Table 2 summarizes the result of the indoor tests. The first column shows the different algorithms and parameters tested. The second column indicates the result of the tests as either success, failed, or close. A close result means that the stopping distance of the EV is too close (based on data gathered from the e-trike drivers) to the obstacle therefore cannot be defined as success nor failure. The third and fourth columns are the maximum and minimum acceleration of the EV as recorded, respectively.

As the indoor tests indicated, out of 13 algorithms: six (6) failed, five (5) were close, and only one (1) was a complete success.

Table 2. Summary of Indoor Tests Results

Algorithm		Outcome	Max Acc (m/s^2)	Min Acc (m/s^2)
DeFuzz		success	0.28	-0.7339
W1CoBa	FN1	close	0.06	-0.2796
	FN2	failed	0.13	0
	NN	failed	0.16	0
W1DeBa	FN1	close	0.05	-0.3297
	FN2	close	0.06	-0.2675
	NN	failed	0.23	0
W2DeBa	FN1	close	0.12	-0.2093
	FN2	close	0.09	-0.2693
	NN	failed	0.48	-0.0293
W1ComB	FN1	failed	0	0
	FN2	close	0.21	-0.1531
	NN	failed	0.14	0

The only complete success recorded was the FL algorithm. This can be expected because of the straightforward mathematical approach of the algorithm of the TSK framework which gave out crisp outputs [11]. As long as the input of the system was defined and was within the parameters, the FL remained robust and predictable [27]. It should also be noted that the acceleration and deceleration values' swing was too large – the EV would accelerate fast and decelerate abruptly providing an unpleasant driving experience.

All NN algorithms (across all variations) failed. This might be due to lack of hidden nodes [30] as this topology gave the largest acceleration but gave the least deceleration in each grouping. Though the topology and parameters converged during training and passed simulation tests, the real-world instrumentation delay might have taken a larger factor in the failure of the implementation of the algorithm.

Both FN algorithms had a three (3) out of four (4) success rates, across different parameters, but are too close than the expected stopping distance. These could also be attributed to instrumentation delay.

It was also observed that the static friction played a huge role in the failed tests. The minimum voltage for the motor to move was 0.25V but it was observed that the EV would not move until it was at 0.5-0.75V. This attributed to a faster starting torque (because of larger current in the excitation coils) and would sometimes hinder the EV from braking before the runway run out. It was also observed that, sometimes when the surface of the obstruction was subjected to intense lighting conditions, the LiDAR sensor does not see the surface. This might be problematic with bare aluminum vehicle surfaces – like jeepneys.

It was then decided to use W1DeBa for the outdoor tests because it has the highest success rate. W2DeBa has the same performance with W1DeBa as expected because they had the same hyperparameters setting: Decaying learning rate with Batch processing. The only difference was that W1 has a negative two (-2) to positive two (+2) random starting weights than the W2 which has a negative one (-1) to positive one (+1) random starting weights.

3.2 Outdoor Tests

The autonomous EV buggy was then subjected to real-world road conditions (inside the campus) compared to the indoor tests which had controlled illumination and road conditions. Table 3 summarizes the result of the outdoor tests. Variation in lighting conditions from good to poor was observed due to trees covering the tracks.

Table 3. Summary of outdoor tests results for different algorithms

	T1	T2	T3	T4	T5	T6	T7	T8
DeFuzz	INC TEST success	jerking success	INC TEST success	jerking success				
H1	maintains 800 success	maintains 200 success	maintains 500 success	maintains 500 success	maintains 800 success	CRASHED Failed	0 success	Too near success
H2	maintains 800 success	0 success	0 success	INC TEST INVALID	INC TEST INVALID	0 success	0 success	0 success
NN	INC TEST success	Too near success	0 Failed	0 success	0 Failed	0 Failed	0 success	Stop and Go Prob Failed

For the outdoor tests, besides the success or failure of the tests, notes were incorporated in the table. A "maintains –" was noted for successful tests. A successful test means that the EV stopped at the appropriate stopping distance as indicated by the algorithm. "INC TEST" means that instrumentation errors were noted but is not related to the success of the stopping of the EV (i.e. error in sensor readings). "Jerking" and not being able to "Stop and Go" were also recorded. These problems may or may not be related to the algorithm. It could be an error in the output of the algorithm due to the erratic output values or an error due to the instrumentation problem in the motor. A "too near" and "crashed" notes were also indicated just like in the indoor tests' results. A "0" in the notes would mean nothing significant could be noted but the test ran smoothly.

The FL algorithm's performance in the outdoor tests showed correct stopping and acceleration responses. However, jerking was observed even during standstill. The crisp output values became a liability in the real-world feel of the driver as it lacks fluid transition from one acceleration value to deceleration [11] [31]. The outdoor tests with the FL algorithm were discontinued after the fourth (4th) test due to the instability of the EV's throttle (due to jerking).

The H1 algorithm passed all its runs except for one (1). It should be noted that the performance of the EV due to the algorithm cannot maintain a consistent headway. It can be attributed to sensor reading error or to the algorithm's performance. The driving experience with the H1 was not comfortable. The accelerations were too fast while decelerations were too late resulting to smaller headway clearance after stopping.

The H2 algorithm passed all its test runs. It must be noted that two of its runs were declared as invalid due to NaN output values recorded in the voltage applied to the motor.

The purely NN algorithm passed and failed its tests equally. Jerking was also observed; like in the FL algorithm driving experience. Detailed data showed that the LiDAR detected the obstacle in front of it, but the algorithm kept the vehicle's acceleration. The NN performed well in the simulations but did poorly in the hardware implementation. This could be attributed to slow algorithm response or lack in robustness of the NN due to the optimized weights [11] [30].

3.3 Hardware Adjustments

The dsPIC30F4011 offered a wide range of capabilities for the small package it comes in. However, there are some restrictions to its memory (program and RAM) that was noted during implementation. The microcontroller has a 48k program memory bytes and an SRAM of 2kB. The program code was divided into five (5) files: main program, machine learning function(s), initialization, and the communication for I2C and UART. Each code has a source code .c file and a header .h file.

The final program uploaded to the microcontroller was summarized in Table 4. The second (2nd) column indicates the total memory space each file would need in the microcontroller. The total was roughly 56kB which was well beyond the maximum allowable program memory of 48kB. The program files were rewritten and compiled again and was shown in the 3rd column. Mostly, the revisions were deleting some commented previous versions of part of the codes and

transitioning other for loops to linear algebraic function to make the code leaner. The weights' data type was also changed from a long double to a double, effectively halving the memory needed per array.

Table 4. Program files' memory requirement

Program Files	Program Memory Space	
	Raw Final	Revised Final
1. main.c	15,814	6,928
2. header.h	1,662	1,662
3. FNNfunction.c	18,240	12,663
4. FNNfunctions.h	620	483
5. Initmod.c	6,879	5,058
6. Initmod.h	1,728	1,728
7. I2Cmod.c	3,958	2,132
8. I2Cmod.h	1,045	1,045
9. UARTmod.c	4,391	4,391
10. UARTmod.h	1,128	1,128
Approximated total memory requirement (Bytes)	55,465	37,218

The main function was also revised to help with the jerking while stationary. The jerking movements were tracked back to the crisp outputs of the defuzzification: some small changes in inputs were observed to sometimes produce small outputs. This behavior was beneficial to the EV in order to have a better transition in changing velocity profile while moving. However, it was not ideal for staying put at zero speed because it resulted in somewhat like a hesitation movement. A smoothing function was included during zero speed to have a better driving experience.

The electric hub motor was also considered to be upgraded to a higher power rating from 1kW 60V to a 3kW 60V or 4.5kW 72V high torque motor. This was to address the delay in movement at starting due to static friction between the tires and the road. The weight of the EV and road conditions contributed largely to the torque requirement of the motor. It was observed that increasing the torque of the motors helped with finer velocity controls but also increased the EV's weight due to the batteries needed to be added and other accessories. This can be researched further to optimize the performance of the EV.

IV. CONCLUSION

The research was able to successfully implement four (4) different algorithms into an EV and observe its performances at different hyperparameters variations. The tests were done in an indoor track with controlled lighting and good road conditions, and in an outdoor road. LiDAR, motors with built-in hall sensors, and optical flow meter were used to measure headway clearance and velocity. The EV was retrofitted with a dsPIC30F4011 microcontroller for processing and control.

A purely FL, one (1) hidden layer FN, two (2) hidden layer FN, and purely NN algorithms were implemented for the EV's stop-and-go autonomous control. It was determined that four (4) parameter settings were to be tested after clearing computer simulations: W1CoBa, W1DeBa, W2DeBa, and W1ComB.

A total of 13 different algorithms were tested in the indoor tests. Maximum and minimum acceleration (and deceleration) swings were observed. Sensor reading errors and discomfortable driving experience (i.FLe. jerking) were sometimes observed in the test runs. W1DeBa, with the highest success rate of stopping before an obstacle, was chosen to be used for the outdoor tests. The outdoor tests revealed that a FN network with two (2) hidden layers (H2) had better control in maintaining proper headway distance and had a more fluid transition in acceleration and deceleration. The purely FL and purely NN algorithms poorly performed in maintaining headway clearance and consistently jerked. The purely NN algorithm's performance could also be attributed to the instrumentation delay of the hardware. Static friction was also observed to cause hindrances in the test runs because it created more starting torque for the EV therefore making it hard to stop in time given the relatively short track. The FN network with one (1) hidden layer (H1) performed better than the FL and NN algorithms but has trouble maintaining a consistent headway clearance. The addition of another layer, H2 compared to H1, had better driving experience (no jerking and less abrupt acceleration and deceleration). H2 also consistently maintained the recommended headway clearance. Two (2) out of the eight (8) tests in H2 were invalid due to sensor reading errors.

V. ACKNOWLEDGEMENT

The authors acknowledge the financial grant provided by the Engineering Research for Development and Technology of Department of Science and Technology, Philippines.

NOMENCLATURE

Symbol	Description	Unit
Acc	Acceleration; set as the output of the system	$[m/s]^2$
Ba	Batch processing used for training neural networks	
C	Constant Acceleration; a MF for Acc	
CF	Closing Fast; a MF for RV	
CL	Closing; a MF for RV	
Co	Constant learning rates used for training neural networks	
De	Decaying learning rates used for training neural networks	
DSSD	Distance sum of squared difference or Distance divergence; set as the input of the system	[m]
EEEI	Electrical and Electronics Engineering Institute	
EV	Electric vehicle	
FL	Fuzzy logic	
FN	Fuzzy-neural network	
FV	Following vehicle	

Symbol	Description	Unit
H1	Fuzzy -neural network with one hidden layer	
H2	Fuzzy -neural network with two hidden layers	
K	Okay; a MF for DSSD	
LA	Light Acceleration; a MF for Acc	
LD	Light Deceleration; a MF for Acc	
LV	Lead Vehicle	
mB	mini-batch processing used for training neural networks	
MF	Membership function	
MTC	Much Too Close; a MF for DSSD	
MTF	Much Too Far; a MF for DSSD	
NN	Neural network	
OF	Opening Fast; a MF for RV	
OP	Opening; a MF for RV	
RV	Relative velocity; set as the input of the system	[m/s]
SA	Strong Acceleration; a MF for Acc	
SD	Strong Deceleration; a MF for Acc	
TC	Too Close; a MF for DSSD	
TF	Too Far; a MF for DSSD	
TSK	Takagi-Sugeno-Kang; an inference system used for fuzzy logic controllers	
UPD	University of the Philippines Diliman	
W1	Random weights using -1 to +1 used for initializing neural networks	
W2	Random weights using -2 to +2 used for initializing neural networks	
Z	Match; a MF for RV	

REFERENCES

[1] Brackstone, M. & McDonald, M., 1999. Car-following: a historical review. Transportation Research Part F: Traffic Psychology and Behavior. 2(4): p. 181–196.

[2] Ming, Q., 1997. Sliding mode controller design for ABS system [Masteral]. Virginia Polytechnic Institute and State University (Va).

[3] Gazis, D., Herman, R. & Rothery, R., 1961. Nonlinear follow-the-leader models of traffic flow. Informs Operations Research. 9(4): pp. 545-567.

[4] Gipps, P., 1981. A behavioural car-following model for computer simulation. Transportation Research Part B: Methodological. 15(2): pp. 105–111.

[5] Bando, M. et al., 1995. Dynamical model of traffic congestion and numerical simulation. Physical Review E (Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics). 51(2): pp. 1035-1042.

[6] Nagel, K. & Schreckenberg, M., 1992. A cellular automaton model for freeway traffic. Journal de Physique I, IEDP Sciences. 2 (12): pp.2221-2229.

[7] Newell, G., 2002. A simplified car-following theory: a lower order model. Transportation Research Part B: Methodological. 36(3): pp. 195–205.

[8] Ma, X., 2006. A neural-fuzzy framework for modeling car-following behavior. 2006 IEEE International Conference on Systems, Man and Cybernetics. Taipei, Taiwan.

[9] Ma, T. & Abdulhai, B., 2001. Genetic algorithm-based combinatorial parametric optimization for the calibration of microscopic traffic simulation models. IEEE Transactions on Intelligent Transportation Systems. Oakland, Ca, USA.

[10] Maurya, A. K. & Bokare, P. S., 2012. Study of deceleration behaviour of different vehicle types. International Journal for Traffic and Transport Engineering. 2(3): pp. 253-270.
[11] Tanaka, M., 2013. Development of various artificial neural network car-following models with converted data sets by a self organizing neural network. Journal of the Eastern Asia Society for Transportation Studies. Volume 10: pp. 1614-1630.
[12] Wang, H., Liu, Z. & Wang, H., 2013. Rear-end and lane changing collisions avoidance theoretical models of the multi lane freeway in disaster weather. Intelligent and Integrated Sustainable Multimodal Transportation Systems Proc. from the 13th COTA International Conference of Transportation Professionals. pp. 70-81.
[13] Dehai, C. & Yidong, W., 2016. A Control Method for Work State of Hybrid Vehicle Driving System Based on Fuzzy Neural Network. 2016 Eighth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). Macau, China.
[14] Sarkodie-Gyan, T. & Willumeit, H., 1995. Neuro-fuzzy based autonomous vehicles. 1995 IEEE International Conference on Fuzzy Systems. Yokohama, Japan.
[15] Rongguang, C., Chunsheng, L., Xia, M. & Yongguang, Y., 2008. The application of fuzzy-neural network on control strategy of hybrid vehicles. 2008 27th Chinese Control Conference. Kunming, China.
[16] Tiflour, B., Boukhnifer, M., Hafaifa, A. & Tanougast, C., 2021. An Optimal Fuzzy Logic Control for a Fuel Cell Hybrid Electric Vehicle Based on Particle Swarm and Advisor. 2021 IEEE Green Technologies Conference (GreenTech). Denver, CO, USA.
[17] Karray, A., Njah, M., Feki, M. & Jallouli, M., 2016. Intelligent Mobile Manipulator Navigation using Hybrid Adaptive-Fuzzy COntroller. Computers and Electrical Engineering. Volume 56: pp. 773-783.
[18] Peng, T. et al., 2020. A New Safe Lane-Change Trajectory model and Collision Avoidance Control Method for Automatic Driving Vehicles. Expert Systems with Applications. Volume 141.
[19] Zhang, S., Abdel-Aty, M., Cai, Q. & Ugan, J., 2020. Prediction of Pedestrian-vehicle conflicts at signalized intersections based on long short-term memory neural network. Accident Analysis and Prevention, Volume 148.
[20] Wu, S., Zhu, E., Ren, H. & Lei, Z., 2008. Control of Four-Wheel-Steering Vehicle Using GA Fuzzy Neural Network. 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA). Changsha, China.
[21] Zhang, Y. & Sheng, G., 2019. Design of Fuzzy Neural Network PID Controller for Hypersonic Vehicle. 2019 Chinese Automation Congress (CAC). Hangzhou, China.
[22] Cai, L., Rad, A., Chan, W. & Ho, M., 2003. A neural-fuzzy controller for intelligent cruise control of vehicle in highways. Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems. Shanghai, China.
[23] de Campos Souza, P. V., 2020. Fuzzy Neural Networks and Neuro-Fuzzy Networks: A review the main techniques and applications used in the literature. Applied Soft Computing Journal. Volume 92. 106275.
[24] Farahbod, F. & Eftekhari, M., 2012. Comparison of different T-norm operators in classification problems. International Journal of Fuzzy Logic Systems. 2(3).
[25] Kukolj, D., 2002. Design of adaptive Takagi–Sugeno–Kang fuzzy models. Applied Soft Computing. 2(2): pp. 89-103.
[26] Rezaee, B. & Zarandi, M. F., 2010. Data-driven fuzzy modeling for Takagi-Sugeno-Kang fuzzy system. Information Sciences: an International Journal. 180(2): pp. 241-255.
[27] Quek, C., Pasquier, M. & Lim, B., 2009. A novel self-organizing fuzzy rule-based system for modelling traffic flow behaviour. Expert Systems with Applications. 36(10): pp. 12167–12178.
[28] Ramoso, J. & Ramos, Jr., M., 2016. Comparative Study of Different Fuzzy-Neural Configurations for Autonomous Vehicle Following Algorithm. IEEE International Conference on Control System, Computing and Engineering. Penang, Malaysia.
[29] Microchip, 2010. dsPIC30F4011/4012 Data Sheet, s.l.: Microchip Technology Inc..
[30] Jang, J., Sun, C. & Mizutani, E., 1997. Neuro-fuzzy and Soft Computing: A computational approach to learning and machine learning. Pearson.
[31] Sato, T. & Akamatsu, M., 2012. Understanding driver car-following behavior using a fuzzy logic car-following model. Fuzzy Logic - Algorithms, Techniques and Implementations. InTech: pp. 265-282.