A Deep Learning Part-diagnosis Platform(DLPP) based on an In-vehicle On-board gateway for an Autonomous Vehicle

KyungDeuk Kim[1], SuRak Son[1], YiNa Jeong[1] and ByungKwan Lee[1*]
[1] Department of Computer Engineering, College of Engineering, Catholic Kwandong University
Gangneung-si, Gangwon-do, 25601 - Korea
[e-mail: bklee@cku.ac.kr]
*Corresponding author: ByungKwan Lee

Received March 4, 2019; revised May 6, 2019; accepted June 6, 2019; published August 31, 2019

Abstract

Autonomous driving technology is divided into 0~5 levels. Of these, Level 5 is a fully autonomous vehicle that does not require a person to drive at all. The automobile industry has been trying to develop Level 5 to satisfy safety, but commercialization has not yet been achieved. In order to commercialize autonomous unmanned vehicles, there are several problems to be solved for driving safety. To solve one of these, this paper proposes ‘A Deep Learning Part-diagnosis Platform(DLPP) based on an In-vehicle On-board gateway for an Autonomous Vehicle` that diagnoses not only the parts of a vehicle and the sensors belonging to the parts, but also the influence upon other parts when a certain fault happens. The DLPP consists of an In-vehicle On-board gateway(IOG) and a Part Self-diagnosis Module(PSM). Though an existing vehicle gateway was used for the translation of messages happening in a vehicle, the IOG not only has the translation function of an existing gateway but also judges whether a fault happened in a sensor or parts by using a Loopback. The payloads which are used to judge a sensor as normal in the IOG is transferred to the PSM for self-diagnosis. The Part Self-diagnosis Module(PSM) diagnoses parts itself by using the payloads transferred from the IOG. Because the PSM is designed based on an LSTM algorithm, it diagnoses a vehicle's fault by considering the correlation between previous diagnosis result and current measured parts data.

Keywords: OBD-Ⅱ, Sensor Fault Decision, Self-diagnosis, Loopback, On-board gateway

A preliminary version of this paper was presented at ICONI 2018, and was selected as an outstanding paper. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2018R1A2B6007710).

http://doi.org/10.3837/tiis.2019.08.017 **ISSN : 1976-7277**

1. Introduction

Conventional driving vehicles impose not only much cost such as fuel, depreciation and insurance but also a variety of external factors such as jaywalking, failed brakes, road factors, and environmental factors on the drivers. Autonomous driving vehicles have the power to significantly diminish both much cost and externalities imposed on the drivers.

There is an increasing interest in autonomous vehicles around the world, and the need is also being emphasized. Autonomous vehicles have many in-vehicle sensors, lidars and stereo-cameras to create a lot of sensor and vision data. The data is executed by computers and IoT systems safely transfer important information to a cloud. The Autonomous driving platform in the cloud has a Deep Learning system which judges an accurate driving decision-making. The system has a connection to the DB with previous driving history. Autonomous vehicles now decides to do something in general or specific circumstances while driving [1].

Companies around the world are increasingly interested in Autonomous Vehicles, so they are aiming at faster commercialization of autonomous vehicles. In order to accelerate its commercialization, autonomous driving requires no external threats and no internal part fault because of safety. They have fallen sharply due to the development of high performance sensors, software and AI. However, there is a great lack of research on the part diagnosis of an autonomous vehicle. An autonomous vehicle consists of advanced sensors and high performance Graphics Processing Units (GPU), Electronic Control Unit(ECU), On-Board Diagnostics (OBD), Artificial Intelligence (AI), etc. In autonomous vehicles, sensors are used for various purposes such as position recognition, object recognition, etc., and the usage of sensor is increasing by using fusion of sensor. However, because sensors are fragile components, the self-diagnosis method so far has not been able to diagnose all the sensors of the vehicle. Many problems arise because all the sensors of the vehicle can not be diagnosed. First, it is difficult to grasp the causes of failures in detail. Second, it is difficult to determine the cause of the failure.

To solve one of these, this paper proposes a Deep Learning Part-diagnosis Platform(DLPP) based on an In-vehicle On-board gateway for an Autonomous Vehicle. It diagnoses not only the parts of a vehicle and the sensors belonging to the parts, but also the influence upon other parts when a certain fault happens. It consists of an In-vehicle On-board gateway(IOG) and a Part Self-diagnosis Module(PSM). Though an existing vehicle gateway was used for the translation of messages happening in a vehicle, the IOG not only has the translation function of an existing gateway but also judges whether a fault happened in a sensor or parts by using a Loopback. The payloads which are used to judge a sensor as normal in the IOG is transferred to the PSM for self-diagnosis. The Part Self-diagnosis Module(PSM) diagnoses parts itself by using the payloads transferred from the IOG. Because the PSM is designed based on an LSTM algorithm, it diagnoses a vehicle's fault by considering the correlation between the previous diagnosis result and the data of current measured parts.

The DLPP for an Autonomous Vehicle proposed in this paper is organized as follows. Previous related works will be presented in Section 2. After this, the main DLPP of this paper will be described in Section 3. The implementation details and the results are discussed in Section 4. Finally, the conclusions and future works are given.

2. Related Works

2.1 Vehicle diagnosis

In [2], the 3σ multi-level screening fault-diagnosis uses Gaussian distribution on determining the fault probability of the battery cell terminal voltage. For the statistical fault-analysis of a lot of electrical cars, a neural network is used to model big sample statistical law and fit. Applying the neural network algorithm combines the single car's fault-diagnosis results with big sample statistical regulation, constructs a more complete battery system fault-diagnosis method, and makes a corresponding analysis between the statistical result and an actual vehicle.

In [3], the neural expert system uses a hybrid data mining technique so that it can efficiently and accurately mine a lot of data in the vehicle database which has previous history. The hybrid approach is made by combining two or more data mining techniques and is commonly used to maximize the accuracy of a classifier. It can make a generalization of the data in the DB with the learning ability of a neural network,

In [4], the system works in three phases. The first phase presents two steering controllers using the sliding mode and the switched H controllers, the second phase describes an unknown input sliding mode observer, and the last phase presents an approach to the diagnosis of critical driving situations. All existing conditions are established using the Lyapunov approach.

2.2 Loop-Back

In [5], Three loopback methods are used to address the need to consider dual link failures and to handle these failures. The first two methods requires two edge-disjoint backup paths computed for each link for rerouting traffic when a pair of links fails. The methods require the identification of the failed links before recovery is completed. The last method requires the pre-computation of a single backup path and does not require link identification before recovery. An algorithm that pre-computes backup paths for links in order to tolerate double-link failures is then presented.

In [6], a simple frequency domain joint transmitter and receiver I/Q imbalance estimation method is used for self-calibration of such wideband multichannel transceivers. Using two frequency domain training signals and a phase shifter inserted in the transceiver local loopback channel, the transmitter and receiver I/Q imbalances can be estimated separately. The estimation errors are also analyzed and the mean square error lower bounds are derived.

In [7], a loopback linearity testing technique for an ADC/DAC pair is used; the key idea is to raise the effective ADC and DAC resolution by scaling the DAC output. First, during ADC testing, we scale down the DAC output to achieve the required test stimulus resolution and adjust the DAC output offset to cover the ADC full-scale range. Then, for DAC testing, we raise the effective ADC resolution by scaling up the DAC output.

In [8], a cost-effective self-characterization technique is demonstrated that accurately predicts the harmonics of individual mixed-signal circuits by dithering the loopback path to Gaussian noise. A set of scale factors is applied to the dithered root-mean-square noise. The different scale factors yield a corresponding change in the harmonic magnitudes of the loopback responses. Based on this, we derive a precise nonlinear loopback behavior model to quantitatively identify the harmonics of a device under test. The results show that the proposed method can be used for practical characterization.

In [9], a new peer-to-peer streaming system is used that applies Loopback-MDC to a real network named TuBeck. The TuBeck consists of preprocessor, server, peer, and player modules with the support of network infrastructure library. In TuBeck, multimedia sources are preprocessed to multiple descriptions. Each description is divided as a sequence of H.264/AVC chunks by modified JM encoder for real-time streaming. The network infrastructure library provides the nodes (server and peers) with the fundamental functionalities of network connections and communications. Message exchange and streaming control among nodes follow the pre-defined LM protocol. The server, peer, and player modules are functionally multithreaded in order to enhance the system performance.

2.3 LSTM

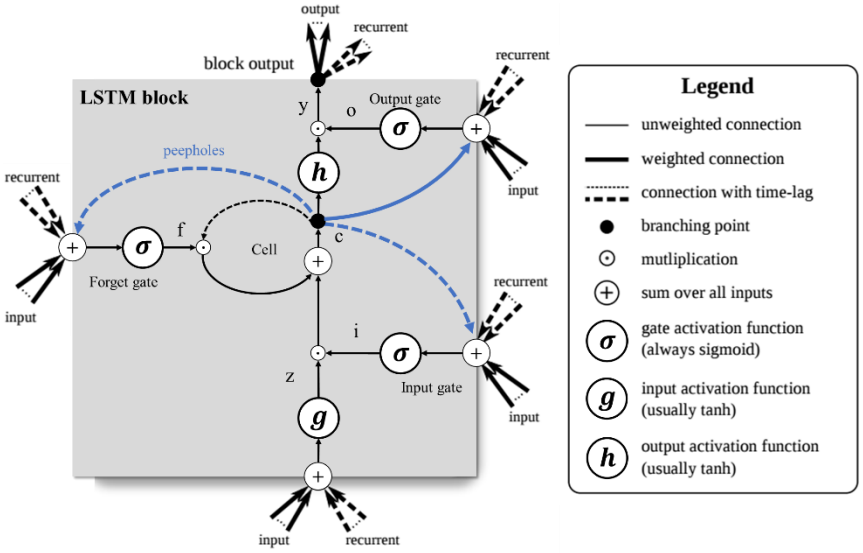


Fig. 1. The composition of a Long Short-Term Memory (LSTM)

Long Short-Term memory (LSTM) networks are a special kind of RNN, capable of learning long-term dependencies and computing both images as single data and speech or video as stream data. In **Fig. 1**, LSTM unit is made up of a cell state, an input gate, an output gate and a forget gate. The cell state is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM and the LSTM gates compute an activation. In brief, the input gate manipulates the scope to which a new value goes into the cell, the forget gate manipulates the scope to which a value stays in the cell and the output gate manipulates the scope to which the value in the cell is used to compute the output activation of the LSTM unit. A forget gate removes the information in the self-recurrent unit and makes space for a new memory. The forget gate does this by multiplying the value of the memory cell by any number between 0(to delete) and 1(to keep). The exact value is determined by the current input and the output of the previous phase. The LSTM model adds the input gate for any new information not to go into the memory cell and the output gate for it to manipulate the output from the memory cell by multiplying the output of the memory cell by any number between 0(no outputs) and 1(to keep output). This output gate may be of use if various memories compete against each other [10].

In [11], a novel model of full-path learning recommendation is used. This model relies on clustering and machine learning techniques. Based on a feature similarity metric on learners,

they first cluster a collection of learners and train a Long Short-Term Memory (LSTM) model in order to predict their learning paths and performance. Personalized learning full-paths are then selected from the results of path prediction. Finally, a suitable learning full-path is recommended specifically to a test learner.

In [12], the SR(Sentence Representations)-LSTM uses 2 hidden layers to address the drawbacks of not being able to catch sufficiently sentimental messages in the existing neural network model for a long time. The 1st layer learns sentence vectors to represent semantics of sentences with an LSTM network, and the 2nd layer translates the relations of sentences into document.

In [13], an attention network for object tracking is used. To construct the proposed attention network for sequential data, we combine Long-Short Term Memory (LSTM) and a residual framework into a Residual LSTM (RLSTM). The LSTM, which learns temporal correlation, is used for a temporal learning of object tracking.

2.4 Sensor

In [14], an RF-DC power conversion system is designed to efficiently convert far-field RF energy to DC voltages at very low received power and voltages. Passive rectifier circuits are designed in a 0.25 mum CMOS technology using floating gate transistors as rectifying diodes. The 36-stage rectifier can rectify input voltages as low as 50 mV with a voltage gain of 6.4 and operates with received power as low as 5.5 muW(22.6 dBm). Optimized for far field, the circuit operates at a distance of 44 m from a 4W EIRP source.

In [15], the design of mobile sensor networks for optimal data collection is addressed. The development is strongly motivated by the application to adaptive ocean sampling for an autonomous ocean observing and prediction system. A performance metric, used to derive optimal paths for the network of mobile sensors, defines the optimal data set as one which minimizes error in a model estimate of the sampled field.

In [16], a wireless wearable system is described that was developed to provide quantitative gait analysis outside the confines of the traditional motion laboratory. The sensor suite includes three orthogonal accelerometers, three orthogonal gyroscopes, four force sensors, two bidirectional bend sensors, two dynamic pressure sensors, as well as electric field height sensors. The "GaitShoe" was built to be worn in any shoe, without interfering with gait and was designed to collect data unobtrusively, in any environment, and over long periods.

3. A Deep Learning Part-diagnosis Platform(DLPP) based on an In-vehicle On-board gateway for an Autonomous Vehicle

3.1 Overview

Fig. 2 shows the composition of ‘A Deep Learning Part-diagnosis Platform(DLPP) based on an In-vehicle On-board gateway for an Autonomous Vehicle`, which is composed of an In-vehicle On-board gateway(IOG) and a Part Self-diagnosis Module(PSM). Though an existing vehicle gateway was used for the translation of messages happening in a vehicle, the IOG not only has the function of an existing gateway but also judges whether a fault happened in a sensor or parts by using a Loopback. The payload which judges a sensor as normal in the IOG is transferred to the PSM for self-diagnosis. The Part Self-diagnosis Module(PSM) diagnoses parts itself by using the payload transferred from the IOG. Because the PSM is designed based on an LSTM algorithm, it diagnoses a vehicle's fault by considering the

correlation between previous diagnosis result and current parts.

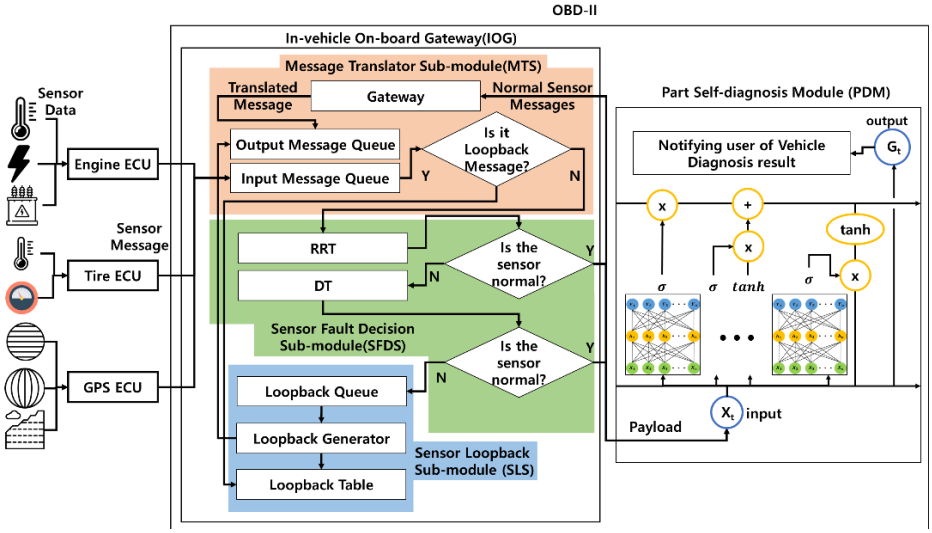


Fig. 2. The composition of the DLPP

3.2 An In-vehicle On-board gateway(IOG) for Message Translation and Sensor Diagnosis

The In-vehicle On-board gateway(IOG) proposed in this paper consists of the 3 sub-modules explained in the following. First, a Message Translator Sub-module(MTS) translates messages for the communication between two other protocols and transfers the translated messages to a destination. Second, a Sensor Fault Decision Sub-module(SFDS) judges whether sensors are a fault by using a vehicle's sensor messages. Third, a Message Loopback Sub-module(MLS) judges whether faulty happened in a sensor or in Parts by using Loopback commands with the sensor messages which the SFDS judged as abnormal. **Fig. 3** shows the composition of the IOG.

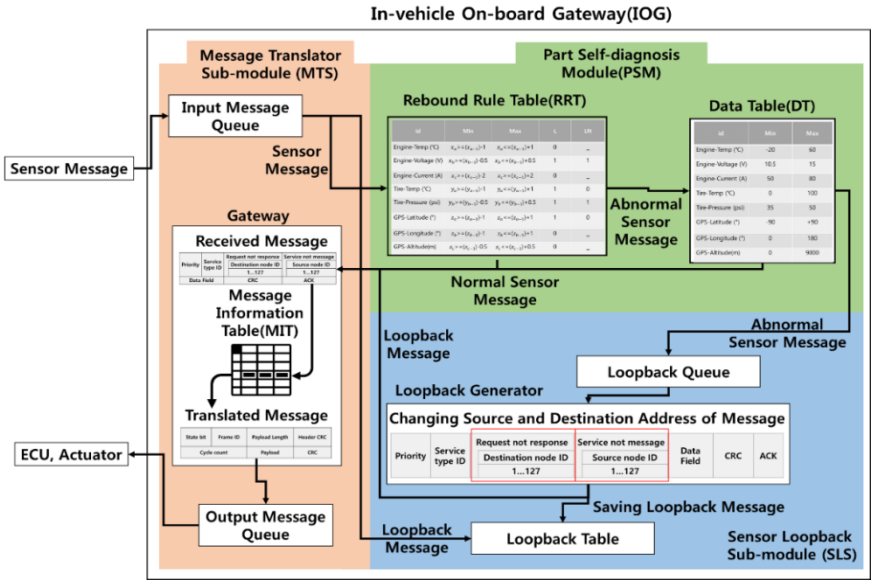


Fig. 3. The composition of the IOG

3.2.1 A Message Translation Sub-module(MTS)

A Message Translation Sub-module(MTS) which performs the role of a gateway consists of an Input Message Queue, an Output Message Queue and a gateway. To begin with, the MTS receives sensor messages from each ECU and stores them in an Input Message Queue. If the source and destination address of the messages are the same, the MTS sends the stored messages as Loopback messages to the SLS [17].

If the source and destination address of the messages are not the same, the MTS sends to the SFDS the sensor messages stored in an Input Message Queue. At this time, the SFDS judges whether the received messages is normal or not. If the message is normal, it is sent to the gateway of the MTS. The MTS extracts only the information necessary for translation from messages and stores the extracted information in the MIT. The MTS translates the stored information into the type of a destination protocol. **Fig. 4** shows that the MTS translates messages into the type of a destination protocol by using the MIT.

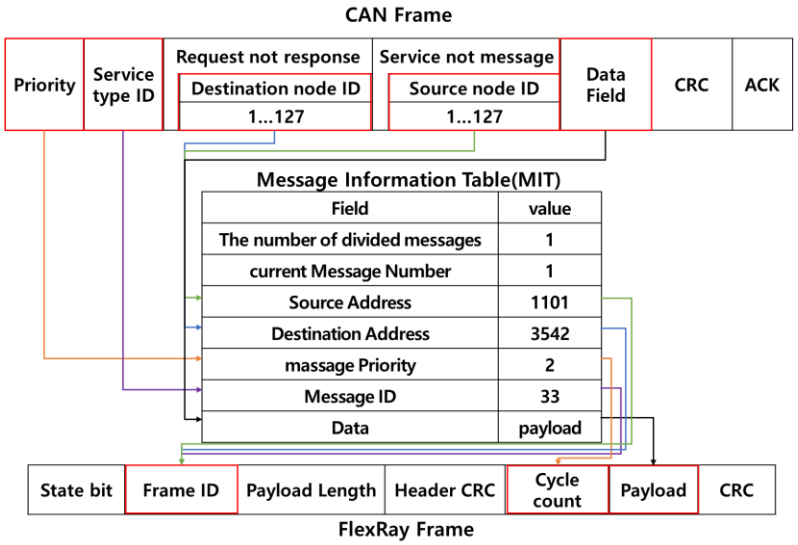


Fig. 4. The gateway translating a CAN message into a FlexRay message

In **Fig. 4**, the fields of the MIT consist of the count of divided messages, current message number, source address, destination address, message priority, message ID and Data. If the size of data before translated is larger than that of data which is being translated currently, the count of divided messages and Current message number fields are used to divide messages. The Destination node ID and Source node ID of a CAN frame are, to begin with, stored in the source and destination address fields of the MIT and then these are stored in the Frame ID of a FlexRay Frame. The Priority and Service type ID of a CAN frame are stored in the Message Priority and Message ID field of the MIT and the Message Priority is translated into a Cycle count of a FlexRay Frame and the Message ID into Frame ID. Because the CRC and ACK of a CAN Frame do not need translation, they are not stored in the MIT. When messages are translated, the State bit, Payload Length, Header CRC and CRC of a Flexray frame are generated newly according to the each message. If the gateway completes message translation, the translated messages are sent to an Output Message Queue. The MTS sends the messages to a destination according to the Priority of messages stored in an Output Message Queue.

3.2.2 A Sensor Fault Decision Sub-module(SFDS)

The SFDS receives a sensor message from the MTS and judges whether its payload is normal. The SFDS uses a Rebound Rule Table(RRT) and a Data Table(DT) to judge the normal value of the payload. **Fig. 5** shows the composition of the RRT and DT.

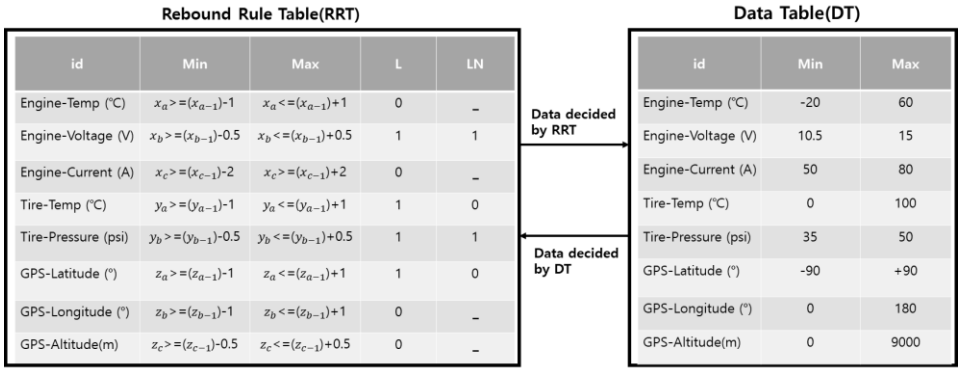


Fig. 5. The composition of the RRT and DT and the corelation between them

The RRT fields consist of ID, MIN, MAX, L and LN. ID field represents sensor ID. MIN and MAX field represents the Max and Min of the current payload which is computed on the basis of the payload received just before. If the just previous payload does not exist, the SFDS computes the Max and Min by using the normal value of sensors. L(Loopback) and LN(Loopback Normality) field represents a flag. L is a flag representing whether the Loopback about a sensor was executed. If L is '0', it means that the Loopback about a sensor was not executed. If L is '1', it means that the Loopback about a sensor was executed. LN represents the result of Loopback when Loopback about a sensor was executed. If LN is '0', it means that the result of the Loopback is an error. If LN is '1', it means that the result of the Loopback is a success. If the L field of the RRT is '0', the value of LN field represents 'NULL'. Algorithm 1 shows the process to judge whether a sensor is normal or not.

Algorithm1. Sensor Fault Decision Sub-module

```
SFD(int sensorID, int data, int Loop[], int RRT[][], int DT[][]){
  //indexing
  const int ID =0, int MIN=1, int MAX=2, int LF=3, int LN=4;
  int n=0, i=0;
  // Finding the ID of the sensor received from RRT
  for(i=0; i<length(RRT); i++){
   if(RRT[i][ID] == sensorID)
     n = i;}
  if(RRT[n][L] == 1 && RRT[n][LN] == 0){ //Abnormal Payload
   end; }
  else{ // Normal Payload
     if(RRT[n][MIN] <= data && data <= RRT[n][MAX]){
        // Sensors that have not yet run loopback
        if(RRT[n][L] == 0){
           if(DT[n][MIN] <= data && data <= DT[n][MAX]){
```

```
            send(data, PSM);
                end;}
              else{
                SLS.LoopBack(sensorID, data, Loop[]);
            RRT[n][L] = Loop[0];
            RRT[n][LN] = Loop[1];
            end;}}
// Sensor that executed loopback
            else{
              RRT[n][L] = 0;
           RRT[n][LN] = Null;
              send(data, PSM);
              end;}}
         else{
         SLS.LoopBack(sensorID, data, Loop[]);
         RRT[n][LF] = Loop[0];
         RRT[n][LN] = Loop[1];
         end;}}}
```

3.2.3 A Sensor Loopback Sub-module(SLS)

The Sensor Loopback Sub-module(SLS) judges whether a sensor is faulty or not actually by using Loopback. If the sensor messages which are judged as abnormal in the SFDS are transferred to the SLS, the SLS stores them in a Loopback Queue. The SLS translates the messages stored in the Loopback Queue into Loopback Messages in sequence. The translated messages are transferred to the Message Queue of the MTS and tored in the Loopback Table. The Loopback Messages stored in the Output Message Queue are transferred to the protocol bus suitable for a message type and then to the SFDS again. If the SFDS received Loopback Messages from outside, they are stored in the Input Message Queue of the MTS like other messages. The MTS transfers to the SLS the message whose source and destination address is IOG. If the SLS receives Loopback Messages from the MTS, it compares the messages stored in the Loopback Table with those received from the MTS. If two kinds of messages are the same, the SLS judges that the sensor is normal but if they are different, the SLS judges that the sensor is not normal. Then the SLS transfers the judgment result to the SFDS.

3.3 A Part Self-diagnosis Module (PSM)

The PSM in the DLPP uses an LSTM algorithm for the self- diagnosis of a vehicle`s parts and the LSTM memorizes the diagnosis result longer than the RNN because of a forget gate. The PSM receives payloads from the IOG steadily and diagnoses the parts by entering the payloads into the LSTM model. In particular, because the PSM computes the correlation between previous diagnosis result and current measured parts` data, it diagnoses the condition of parts accurately. **Fig. 6** shows the simple composition of the PSM, where X represents the input of the LSTM as the set of payload collected from a vehicle. G represents the output of the LSTM and the condition of parts diagnosed in the LSTM. Because t represents specific time, the X_t of **Fig. 6** means the set of payloads used as input at a specific time.

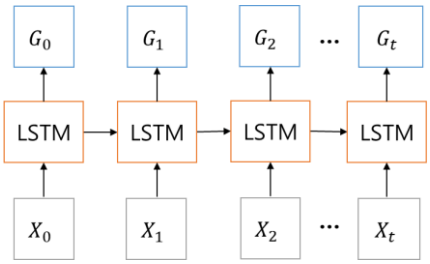


Fig. 6. The composition of the PSM

3.3.1 input and output of PSM

To begin with, the input and output of the PSM has to be defined for the self-diagnosis of a vehicle. The input of the PSM represents all the payloads transferred from the IOG. Because the IOG transfers the payload value 0 of a faulty sensor to the PSM to judge whether the sensor is normal, the payload is not applied to self-diagnosis. The input, *X* of the PSM is expressed like the Formula (1) and the *n* of Formula(1) means the count of in-vehicle sensors.

$$\mathrm{X} = \{x_1, x_2, x_3, x_4, \dots, x_n\} \tag{1}$$

The output of the PSM represents diagnosis result. The condition of parts is decided according to the value of each output node which exists between -1 and 1. If the value is less than -0.4, the PSM judges the part as fault. If the value is greater than -0.4 and less than 0.4, the PSM judges that the part needs to be checked. If the value is greater than 0.4, it judges that the part is normal. The set of an output node, G is expressed like the Formula (2) and m of Formula (2) means the count of vehicle parts.

$$\mathrm{G} = \{g_1, g_2, g_3, g_4, g_5, \dots, g_m\} \tag{2}$$

3.3.2 The Operation process of PSM

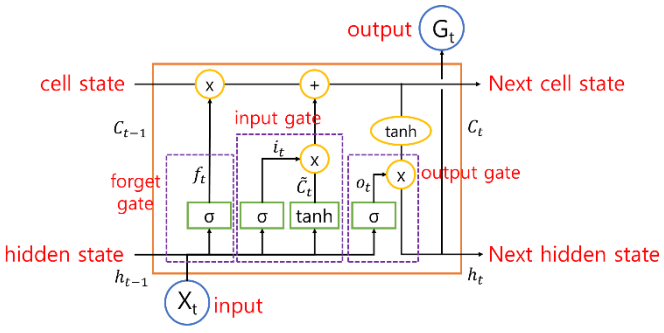


Fig. 7. The Structure of LSTM

Fig. 7 shows the Structure of the LSTM consisting of 2 States and 3 gates. In the **Fig. 7**, black lines represent the direction where data is headed and green squares represent the neural network model used in each gate. The σ and tanh within the green squares mean activation functions. In the Formula(3) and (4), σ represents a sigmoid function and tanh a tanh function.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3}$$

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{4}$$

The yellow circle of **Fig. 7** represents an operator between data. If the data reaches the yellow circle, it is computed with the operator. The Cell State of the LSTM means an area to decide what data to memorize, by comparing previous output with current input and the Hidden State means an area to apply previous output and current input to the neural network model used in each gate. The computation of the LSTM is done in the 3 gates (input, output, and forget).

The forget gate decides what data to memorize among previous data in the Cell state and transfers to the Cell State the value of the forget gate, f_t computed with the Sigmoid Function of Formula(5) in the Hidden State by using previous output and current input.

$$f_t = \sigma(W_f \circ [h_{t-1}, x_t] + b_f) \tag{5}$$

In Formula(5), W_f means weight, h_{t-1} previous output, x_t current input, and b_f the bias of the neural network model. The input gate computes i_t in Formula(6) in the same way as f_t and $\tilde{C}_t$ with the tanh function in Formula(7).

$$i_t = \sigma(W_i \circ [h_{t-1}, x_t] + b_i) \tag{6}$$

$$\tilde{C}_t = \tanh(W_C \circ [h_{t-1}, x_t] + b_C) \tag{7}$$

The result that the i_t is multiplied by the $\tilde{C}_t$ is transferred to the Cell state. The input gate decides whether new data has to be added to the information processed in the forget gate. If the input gate and the forget gate complete computation, the DLPP updates the Cell state. The C_t which the Cell state updated is computed in Formula(8).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{8}$$

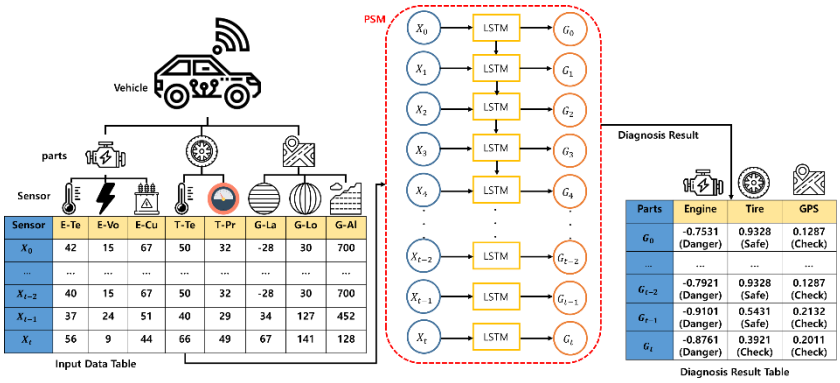
In Formula(8), the $f_t * C_{t-1}$ means the result of the forget gate and decides what data to remove among past results. The $i_t * \tilde{C}_t$ means the result of the input gate and decides what data is add to the data processed in the forget gate.

If the update of the Cell state is completed, the output gate decides what data to output among data of the Cell state. The result that the o_t computed by Formula(9) in the output gate is multiplied by the $\tanh(C_t)$ computed by Formula(10) in the Cell State becomes the final output h_t.

$$o_t = \sigma(W_o \circ [h_{t-1}, x_t] + b_o) \tag{9}$$

$$h_t = o_t * \tanh(C_t) \tag{10}$$

The neural network model used in each gate of the LSTM is learned by using Back-propagation. The PSM compares the previously diagnosed condition of a vehicle with the current diagnosed condition of a vehicle by using the LSTM steadily.



Sensor	E-Te	E-Vo	E-Cu	T-Te	T-Pr	G-La	G-Lo	G-Al
X_0	42	15	67	50	32	-28	30	700
...	...	...	...	...	...	...	...	...
X_{t-2}	40	15	67	50	32	-28	30	700
X_{t-1}	37	24	51	40	29	34	127	452
X_t	56	9	44	66	49	67	141	128

Parts	Engine	Tire	GPS
G_0	-0.7531 (Danger)	0.9328 (Safe)	0.1287 (Check)
...	...	...	...
G_{t-2}	-0.7921 (Danger)	0.9328 (Safe)	0.1287 (Check)
G_{t-1}	-0.9101 (Danger)	0.5431 (Safe)	0.2132 (Check)
G_t	-0.8761 (Danger)	0.3921 (Check)	0.2011 (Check)

Fig. 8. The composition of the Part Self-diagnosis Module (PSM).

Fig. 8 shows that 3 parts are diagnosed and the data of 8 sensors as input data are used in the PSM. The kinds of sensors measuring the condition of an Engine are Temp, Voltage and Current, the kinds of sensors measuring a Tire are Temp and Pressure and the kinds of sensors measuring the condition of a GPS are Latitude, Longitude and Altitude. The left input Data Table shows the payloads of the 8 sensors are entered into the PSM every 1ms. The LSTM model diagnoses a vehicle using the input payloads. The right Diagnosis Result Table shows the diagnosis result of the PSM and the G_0~G_t means the diagnosis result of payloads measured every 1ms. The diagnosis result exists between -1 and 1 representing the condition of parts and the PSM judges the risk of parts according to the diagnosis result. In **Fig. 8**, the G_0 of the Diagnosis Result Table shows the diagnosis result of parts using X_0. Because the diagnosis result of G_0 in an Engine is -0.7531 and it is between -1.0 and -0.4, the PSM judges the Engine as "Danger". Because diagnosis result of G_0 in a Tire is 0.9328 and it is between 0.4 and 0.1, the PSM judges the Tire as "Safe". Finally, because the diagnosis result of G_0 in GPS is 0.1287 and it is between -0.4 and 0.4, the PSM judges the GPS as "Check".

3.3.3 The Neural Networks of PSM

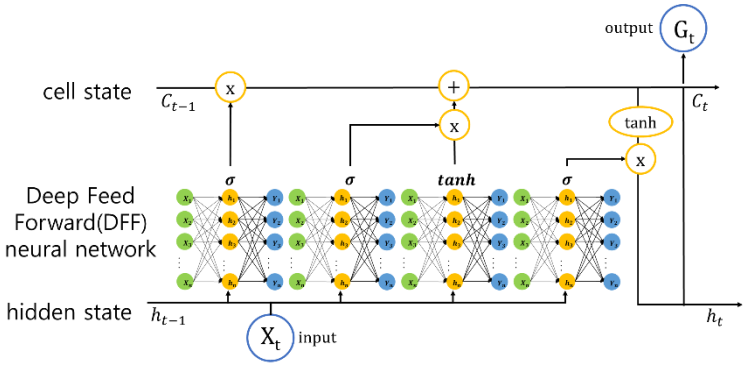


Fig. 9. The Structure of LSTM

Fig. 9 shows that the LSTM consists of 2 States and 4 Deep Feed Forward(DFF) neural network models. All the payloads becomes the input data of each neural network model. 2 States was already discussed in section 3.3.2. The DFF is differently generated according to the count of vehicle sensors, a vehicle model, etc., but DFF's input, hidden layer, weight,

output and learning method have to be designed to generate the DFF. 4 neural network models are designed in the same way, as explained in the following.

First, the DFF uses the result, X_b that b(bias) was added to the input value, X of the PSM. The X_b is represented like Formula (11). The bias of all neural network models is 1 and is adjusted by weight. Because the bias is attached to Formula(11), the count of input nodes is $n + 1$.

$$X_b = \{x_1, x_2, x_3, x_4, \dots x_n, 1\} \tag{11}$$

Second, the DFF uses 4 hidden layers and weights. The count of hidden layer nodes is larger than that of input layer nodes by 15. The DFF computes the value of each node using all the weight from the nodes of input layer to that of output layer. The set of hidden layers, H is represented like Formula (12) and the set of weights, W like Formula(13).

$$H^k = \{\mathrm{h}_1^k, \mathrm{h}_2^k, \mathrm{h}_3^k, \mathrm{h}_4^k, \dots, h_o^k\} \tag{12}$$
$$W^q = \{\mathrm{w}_{1\,1}^q, \mathrm{w}_{1\,2}^q, \mathrm{w}_{1\,3}^q, \mathrm{w}_{1\,4}^q, \dots, w_{l\,p}^q\} \tag{13}$$

In Formula (12), the k which has the value of 1 to 4 means a current hidden layer`s order. The o represents the count of hidden layer nodes. h_4^2 represents the 4th node of the 2nd hidden layer. The size of the o is larger than that of input data by 16. In Formula (13), the q represents what order a current weight is and has a value between 1 and 5. The l represents a node count of a layer where current data is entered and the p represents a node count of the next layer where current data is outputted. $w_{4\,2}^4$ represents the 4th weight connecting the 4th node of the 3rd hidden layer and the 2nd node of the 4th hidden layer.

Third, the output G of the DFF which is a set of output nodes represents the condition of vehicle parts and is expressed like Formula(14).

$$\mathrm{G} = \{g_1, g_2, g_3, g_4, \dots, g_m\} \tag{14}$$

Here, the m means the count of parts that can be diagnosed in the PSM. The values of output nodes get different according to activation functions. If an activation function is a Sigmoid, the output nodes have the value between 0 and 1 and if it is tanh, they have the value between -1 and 1.

Fourth, the learning method of the DFF is explained here. 4 neural network models are learned based on Back-Propagation. To begin with, training input data is entered in the neural network model with initial weight. Whenever the data entered in a input layer goes to the output layer, the data is computed in each layer using an activation function and weight. Each neural network uses 2 activation functions. The 1st activation function is Leaky ReLU which outputs an accurate result despite many hidden layers. Therefore, the activation function is used from an input layer to the last hidden layer. The 2nd activation function is Sigmoid or tanh. Because the ReLU has a drawback that it has hard time analyzing result, a Sigmoid function with the result between 0 and 1 or a tanh function with the result value between -1 and 1 is used between the last hidden layer and an output layer [18]. Formula (15) represents Leaky ReLU. A Sigmoid and a tanh function have already been explained in section 3.3.2.

$$\mathrm{L\_ReLU(x)} = \max(0.01\mathrm{x}, \mathrm{x}) \tag{15}$$

The node values of each layer by using the node values of a previous layer and the weight connected to the nodes have to be computed to find an output value of the DFF. The values of all NETs from an input layer to the 1st hidden layer is computed in the Formula (16). Each value of NET is computed by multiplying the node values of an input layer by weights. And then an activation function is applied to each NET. The node values of the 1st hidden layer is computed by using the Leaky ReLU of Formula (17).

$$NET_{h_i^1} = \sum_{j=1}^{n} x_j \, w_j^1{}_i + \mathrm{b} \tag{16}$$

$$h_i^1 = L\_ReLU(NET_{h_i^1}) \tag{17}$$

The values of all NET from the 2nd hidden layer to the last hidden layer are computed in Formula (16). The node values from the 2nd hidden layer to the last hidden layer are computed in Formula (17).

The NET value between the last hidden layer and an output layer is computed in Formula (18). The node values between the last hidden layer and output layer are computed using different activation functions(Sigmoid, tanh) in Formula(19) and (20).

$$NET_{g_m} = \sum_{j=1}^{n+16} h_j^4 \, w_j^5{}_m + \mathrm{b} \tag{18}$$

$$g_m = \sigma(NET_{g_m}) \tag{19}$$

$$g_m = tanh(NET_{g_m}) \tag{20}$$

If the values of each layer are computed, an Error between the output value of learning data and that of neural network is computed by Formula(21). If Error is less than $\mathrm{Error}_{\max}$, the learning is closed. Here, the $\mathrm{Error}_{\max}$ is a fixed number.

$$\mathrm{Error} = \sum_{i=1}^{m} \frac{(d_i - g_i)^2}{2} \tag{21}$$

Here, d_i is the output value of learning data. If Error is greater than $\mathrm{Error}_{\max}$, an error signal, δ is computed to modify the weight of DFF. δ_{gi} is the error signal computed between the last hidden layer and an output layer. When the activation function is Sigmoid, δ_{g_i} is computed with Formula(22). When the activation function is tanh, δ_{yi} is computed with Formula(23).

$$\delta_{g_i} = (d_i - g_i) g_i (1 - g_i) \tag{22}$$

$$\delta_{g_i} = (d_i - g_i)(1 - g_i)(1 + g_i) \tag{23}$$

If δ_{gi} was computed, then the error signals of hidden layer, $\delta_{h_1} \sim \delta_{h_4}$ have to be computed. The error signal of hidden layer, δ_{h4} is computed with Formula(24) and the rest, $\delta_{h_1} \sim \delta_{h_3}$ are computed with Formula(25).

$$\begin{aligned} &\text{if(L\_ReLU}\left(NET_{h_i^4}\right) == NET_{h_i^4}) \delta_{h_i^4} = \sum_{j=1}^{m} \delta_{g_j} w_i^5{}_j \\ &\text{elseif(L\_ReLU}\left(NET_{h_i^4}\right) == 0.01 * NET_{h_i^4}) \delta_{h_i^4} = 0.01 * \sum_{j=1}^{m} \delta_{g_j} w_i^5{}_j \end{aligned} \tag{24}$$

$$\text{if(L\_ReLU}\left(NET_{h_i^c}\right) == NET_{h_i^c})\delta_{h_i^c} = \sum_{j=1}^{t}\delta_{h_j^{c+1}}w_{i\,j}^{c+1}$$
$$\text{elseif(L\_ReLU}\left(NET_{h_i^c}\right) == 0.01 * NET_{h_i^c})\delta_{h_i^c} = 0.01 * \sum_{j=1}^{t}\delta_{h_j^{c+1}}w_{i\,j}^{c+1} \quad (25)$$

Here, if an error signal δ_{h_3} of the current 3rd hidden layer is computed, t means the count of nodes of the 4th hidden layer corresponding to the next layer. c means a hidden layer number and a range of c is 1 to3. If the error signal of each node is computed, the weight of a neural network is modified through the error signal. The reason why the value of an error signal is different according to the value of L_ReLU is because differential value is changed according to the value of L_ReLU function.

Formula (26) shows that the weight between the last hidden layer and an output layer is modified, Formula(27) shows that the weight from the last hidden layer to the 1st hidden layer is modified and Formula(28) shows that the weight between the 1st hidden layer and an input layer is modified.

$$w_{i\,j}^{5} = w_{i\,j}^{5} + \alpha\delta_{g_j}h_i^4 \quad (26)$$
$$w_{i\,j}^{u+1} = w_{i\,j}^{u+1} + \alpha\delta_{h_j^{u+1}}h_i^u \quad (27)$$
$$w_{i\,j}^{1} = w_{i\,j}^{1} + \alpha\delta_{h_j^1}x_i \quad (28)$$

Here, u represents a hidden layer number and the learning process continues till Error is less than $\text{Error}_{max.}$

4. The performance Analysis

In this section, the performance analysis on the PSM of the DLPP is done. The PSM is analyzed in terms of accuracy and Operation time. Table 1 shows a PC environment used for the performance analysis.

Table 1. The Experimental Environment of the DLPP

CPU	GPU	RAM	OS
Intel i5-7400	Geforce GTX 1050	SAMSUNG DDR4 8GB	Window 10 Education

Under the same experimental condition, the CNN, MLP and PSM performance on Operation time and accuracy is analyzed in this section. The 1st experiment shows that the Operation time of 3 neural network models is measured according to the count of test data sets and **Fig. 10** shows the Operation time as the count of training data sets gets increased from 200 to 900.

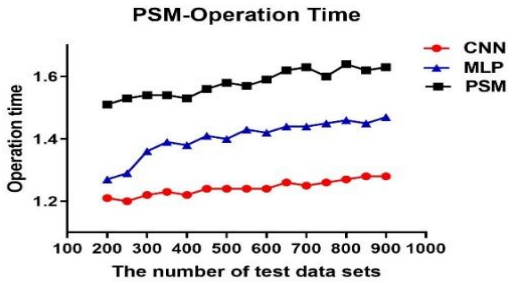


Fig. 10. Operlation time performance analysis of PSM

In case of the CNN, The shortest Operation time is 1.20ms when 250 sets are used and the longest Operation time is 1.28ms when 900 sets are used. Therefore, the average Operation time of the CNN is about 1.24ms. In case of the MLP, the shortest Operation time is 1.27ms when 200 sets are used and the longest Operation time is 1.47ms when 900 sets are used. Therefore, the average Operation time of the MLP is about 1.47ms. In case of the PSM, The shortest Operation time is 1.51ms when 250 sets are used and the longest Operation time is 1.63ms when 900 sets are used. Therefore, the average Operation time of the PSM is about 1.57ms.Therefore, the PSM is slower by about 0.19ms than the CNN and 0.1ms than the MLP in term of the Operation time.

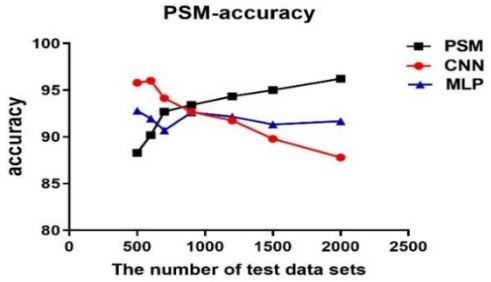


Fig. 11. Accuracy performance analysis of PSM

The 2nd experiment compares 3 neural network models in terms of accuracy as test data sets get increased from 500 to 2000 in **Fig. 11**. In case of the CNN, the accuracy is 95.8% when 500 sets are used, 94.14% when 700 sets are used, 91.75% when 1200 sets are used and 87.8% when 2,000 sets are used. In case of the MLP, the accuracy is 92.8% when 500 sets are used, 90.96% when 700 sets are used, 92.15% when 1200 sets are used and 91.68% when 2,000 sets are used. In case of the PSM, the accuracy is 88.30.8% when 500 sets are used, 92.7% when 700 sets are used, 94.35% when 1200 sets are used and 96.23% when 2,000 sets are used.

When 500 sets are used, the accuracy of the PSM is lower by 7% than that of the CNN and by 4% than that of the MLP. However, when more sets than 900 are used, the accuracy of the PSM gets higher than that of the CNN and MLP. When 2000 test sets are used, the accuracy of the PSM is higher by about 9% than that of the CNN by about 5% than that of the MLP. This experiment shows that the accuracy of the PSM gets more increased as the count of data sets gets larger.

Finally, the CNN has about 0.2ms faster operation time and about 7% lower accuracy than the PSM and the MLP has about 0.1ms faster operation time and about 4% lower accuracy than the PSM. Because there is little difference among them in operation time, the PSM with higher accuracy are more suitable for vehicle diagnosis than these two neural network models.

5. Conclusion

This paper proposes a Deep Learning Part-diagnosis Platform(DLPP) based on an In-vehicle On-board gateway for an Autonomous Vehicle. It diagnoses not only the parts of a vehicle and the sensors belonging to the parts, but also the influence upon other parts when a certain fault happens. The DLPP consists of an In-vehicle On-board gateway(IOG) and a Part Self-diagnosis Module(PSM). Though an existing vehicle gateway was used for the translation of messages happening in a vehicle, the IOG not only has the translation function of an existing gateway but also judges whether a fault happened in a sensor or parts by using a Loopback. The payloads which are used to judge a sensor as normal in the IOG is transferred to the PSM for self-diagnosis. The Part Self-diagnosis Module(PSM) diagnoses parts itself by using the payloads transferred from the IOG.

The experiments of the DLPP were done to verify the efficiency. The PSM took longer by about 0.2ms than the CNN and about 0.1ms than the MLP in Operation time, but improved the accuracy higher by about 9% than the CNN and about 5% than the MLP. In the experiment, because the difference value of the Operation time is very small, it can be ignored. The PSM can be selected in terms of accuracy, compared to the CNN and MLP. Therefore, the DLPP can do a detailed self-diagnoses between parts and sensors and diagnose the influence upon other parts because of a part`s fault. In addition, If a fault happens to only a sensor, the DLPP has an advantage that just the sensor belonging to a part can be replaced, not to replace the part. Because of this advantage, the DLPP has an economic effect and prevents an accident in advance.

References

[1] Suhasini Gadam, “Artificial Intelligence and Autonomous Vehicles,” *Data Driven Investor*, 2018. Article (CrossRef Link)

[2] Zhao Yang., Liu Peng., Wang Zhenpo., Hong Jichao., “Electric Vehicle Battery Fault Diagnosis Based on Statistical Method,” *Energy Procedia*, Vol. 105, pp.2366-2371, 2017. Article (CrossRef Link)

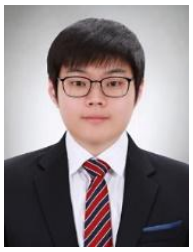
[3] Fong A.C.M., Hui S.C., “Neural expert system for vehicle fault diagnosis via the WWW,” *Computational Web Intelligence*, pp..169-181, 2004. Article (CrossRef Link)

[4] Menhour L., Charara A., Lechner D., "Steering vehicle control and road bank angle estimation: application for diagnosis of vehicle limits in bend," *Inderscience Publishers*, Vol. 12, No. 4, pp334-366, 2014. Article (CrossRef Link)

[5] Guo Dingfei., Zhong Maiying., Ji Hongquan,, Liu Yang., Yang Rui., “A hybrid feature model and deep learning based fault diagnosis for unmanned aerial vehicle sensors,” *Neurocomputing*, Vol. 319, pp.155-163, 2018. Article (CrossRef Link)

[6] Hongsik Choi., Suresh Subramaniam., Hyeong-Ah Choi., "Loopback recovery from double-link failures in optical mesh networks," *IEEE/ACM Transactions on Networking*, Vol. 12, Issue 6, pp1119-1130, 2004. Article (CrossRef Link)
[7] Xiaojing Huang., Y. Jay Guo., Jian A. Zhang., "Transceiver I/Q Imbalance Self-Calibration With Phase-Shifted Local Loopback for Multichannel Microwave Backhaul," *IEEE Transactions on Wireless Communications*, Vol. 15, pp.7657-7669, 2016. Article (CrossRef Link)
[8] Xuan-Lun Huang., Jiun-Lang Huang., "ADC/DAC Loopback Linearity Testing by DAC Output Offsetting and Scaling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 19, pp.1765-1774, 2011.
[9] Byoungho Kim., "Dithering Loopback-Based Prediction Technique for Mixed-Signal Embedded System Specifications," *IEEE Transactions on Circuits and Systems II-Express Briefs*, Vol. 63, pp.121-125, 2016. Article (CrossRef Link)
[10] Long Short-Term Memory (LSTM). https://developer.nvidia.com/discover/lstm
[11] Zhou Yuwen., Huang Changqin., Hu Qintai., Zhu Jia,, Tang Yong., "Personalized learning full-path recommendation model based on LSTM neural networks," *Information Sciences*, Vol. 444, pp135-152, 2018. Article (CrossRef Link)
[12] Rao Guozheng., Huang Weihang., Feng Zhiyong,, Cong Qiong., "LSTM with sentence representations for document-level sentiment classification," *Neurocomputing*, Vol. 308, pp.49-57, 2018. Article (CrossRef Link)
[13] Hong-In Kim., Rae-Hong Park., "Residual LSTM Attention Network for Object Tracking," *IEEE Signal Processing Letters*, Vol. 25, pp.1029-1033, 2018. Article (CrossRef Link)
[14] T. Le., K. Mayaram., T. Fiez., "Efficient Far-Field Radio Frequency Energy Harvesting for Passively Powered Sensor Networks," *IEEE Journal of Solid-State Circuits*, Vol. 43, pp.1287-1302, 2008. Article (CrossRef Link)
[15] N.E. Leonard., D.A. Paley., F. Lekien., R. Sepulchre., D.M. Fratantoni., R.E. Davis., "Collective Motion, Sensor Networks, and Ocean Sampling," *Proceedings of the IEEE*, Vol. 95, pp.48-74, 2007. Article (CrossRef Link)
[16] S. Bamberg., A.Y. Benbasat., D.M. Scarborough., D.E. Krebs., J.A. Paradiso., "Gait Analysis Using a Shoe-Integrated Wireless Sensor System," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 12, Issue 4, pp..413-423, 2008. Article (CrossRef Link)
[17] Y.N. Jeong., S.R. Son., E.H. Jeong., B.K. Lee., "A Design of a Lightweight In-Vehicle Edge Gateway for the Self-Diagnosis of an Autonomous Vehicle," *Applied Sciences*, Vol. 8, 2018. Article (CrossRef Link)
[18] Y.N. Jeong., S.R. Son., B.K. Lee., "The Lightweight Autonomous Vehicle Self-Diagnosis (LAVS) Using Machine Learning Based on Sensors and Multi-Protocol IoT Gateway," *Sensors*, Vol. 19, 2019. Article (CrossRef Link)

KyungDeuk Kim received his B.Eng degree from Catholic Kwandong University in 2018. His interests are Autonomous vehicle, IoT, network, and Big data. Email: Chamsol93@naver.com



SuRak Son received his B.Eng degree from Kwandong University in 2018. He is currently working towards a master in Computer Science from Catholic Kwandong University. Her current research interests are Artificial neural network, Autonomous vehicle, and Network security. Email: sonsur@naver.com

YiNa Jeong received her B.Eng. degree from Kwandong University in 2011, the M.Eng. degree in Computer Science from Kwan-Dong University in 2012. She is currently working towards a doctorate in Computer Science from Catholic Kwandong University. Her current research interests are Sensor Network, IT security, and Network security Email: upinus07@nate.com

ByungKwan Lee received his B.S. degree from Pusan National University in 1979, the M.S. degree in Computer Science from Chung-Ang University in 1986 and the ph.D. degree in Computer Science from Chung-Ang University in 1990 in Korea. He has been a professor of Computer Science at Catholic Kwandong University in Korea since 1988. He was a visiting professor at Saginaw Valley State university, Michigan, USA during 2000~2001. He is a permanent member of the KISS and KIPS. His current research interests are network security, Wireless Sensor Network Security, Internet of Things and Big Data. Email: bklee@cku.ac.kr