Comparison of Intelligent Methods of SOC Estimation for Battery of Photovoltaic System

Tae-Hyun Cho[1], Hye-Rin Hwang[2], Jong-Hyun Lee[3], In-Soo Lee[4, *]
School of Electronics Engineering
Kyungpook National University
Daegu, Korea

***Abstract*—It is essential to estimate the state of charge (SOC) of lead-acid batteries to improve the stability and reliability of photovoltaic systems. In this paper, we propose SOC estimation methods for a lead-acid battery using a feed-forward neural network (FFNN) and a recurrent neural network (RNN) with a gradient descent (GD), a levenberg–marquardt (LM), and a scaled conjugate gradient (SCG). Additionally, an adaptive neuro-fuzzy inference system (ANFIS) with a hybrid method was proposed. The voltage and current are used as input data of neural networks to estimate the battery SOC. Experimental results show that the RNN with LM has the best performance for the mean squared error, but the ANFIS has the highest convergence speed.**

***Keywords*—Lead-acid battery; SOC; FFNN; RNN; ANFIS; gradient descent; levenberg-marquardt; scaled conjugate gradient**

I. INTRODUCTION

Today, several environmental issues exist, including the depletion of fossil fuels and the dangers of nuclear power generation. For these reasons, the application of renewable energy has increased, and research on solar power has been actively conducted. Photovoltaic (PV) systems can be categorized into grid-connected systems and stand-alone power systems, depending on whether the system is connected to an electrical power-generation system. In particular, in the stand-alone power system, the demand for solar streetlights is urgently increasing, including low-power light-emitting diode (LED) lamps that are used to replace conventional halogen security lighting [1].

The solar streetlight system consists of solar-panel modules that convert solar energy into electricity, a secondary battery that stores the developed power, and a stand-alone (off-grid) system. The stand-alone system has the following advantages: 1) it does not require electric power installation, because the commercial power is not connected, and 2) it has a CO_2-reduction effect because it is operated by PV power generation.

The operation time of the solar streetlight only depends on a secondary battery. Days when sunshine time is less than 0.1 hour are considered sunless days, and solar streetlights should be guaranteed to operate for more than three sunless days.

It is essential to estimate the state of charge (SOC) of lead-acid batteries in real time for the following reasons. First, we must avoid reliance on the initial SOC of the battery. This allows more effective control of the power consumption of LED lamps. Second, the lead-acid battery is the most commonly used energy-storage device for PV systems. According to some researchers, lead-acid batteries will survive in the future.

A battery is a device that generates electrical energy through a chemical reaction, and it has nonlinear characteristics in response to parameters such as the ambient temperature, internal resistance, and capacitance. For these reasons, it is very difficult to estimate the SOC of a battery correctly.

There are many methods for estimating battery SOC [2]. The internal-impedance method estimates the SOC by measuring the internal-impedance change according to the charging and discharging of the battery. However, it is difficult to apply in a state where the cell is reacting, because it is very sensitive. The kalman-filter method is difficult to apply because of the complexity of the parameters and algorithms. The current-integration (CI) method [3] involves subtracting the initial SOC value by integrating the actual charge and discharge current. However, this method cannot estimate the initial SOC, and because of the accumulated errors of the leakage current and current sensing over time, accurate SOC estimation is impossible. The open-circuit voltage (OCV) [4] method involves measuring the voltage in the no-load state. However, this method is difficult to apply to real-time systems because it uses the measured OCV at the chemical equilibrium inside the battery.

Neural networks have proven to be a promising paradigm for intelligent systems. They have been trained to perform complex functions in various fields, such as pattern recognition, identification, and classification [5]. Their ability to learn complex nonlinear input/output relationships, their use of sequential training procedures, and their adaptability to data are three outstanding characteristics of neural networks. Some popular modules of neural networks have shown abilities of associative memory and learning [6-8]. To allow the network to perform a specific classification and clustering task efficiently, the learning process comprises updating the network architecture and modifying the weights between the neurons. The neural network can efficiently model a variety of input and output relationships. Compared with procedural models, it has the advantage of a shorter execution time [9, 10].

In this paper, SOC estimation methods for a lead-acid battery using a feed-forward neural network (FFNN), a

*Corresponding Author.

recurrent neural network (RNN), and an adaptive neuro-fuzzy inference system (ANFIS) were proposed. The FFNN and RNN models were applied with three different training methods: gradient descent (GD), scaled conjugate gradient (SCG), and levenberg–marquardt (LM), under the same network configuration. The ANFIS model was applied with a hybrid training method. Also, we compared the performance of the different networks.

In Section II, research on battery SOC estimation is introduced. In Section III, neural network based battery SOC estimation methods were proposed. In Section IV, experimental results for the FFNN, RNN, and ANFIS models were proposed. Finally, the paper is concluded in Section V.

II. Study on Battery SOC Estimation

A. Open Circuit Voltage Method

The open circuit voltage (OCV) method is widely known as a battery SOC estimation method. The OCV method is very accurate in estimating the SOC when the lead-acid battery reaches the stabilization phase. In addition, data must be constructed through SOC-OCV relationship experiments. Although estimating the SOC from a battery is one of the most effective methods for estimating the SOC using OCV, this method requires a condition in which the circuit is opened or no current flows, and it takes time to wait for the battery to stabilize internally. For this reason, this method is difficult to use in real-time estimation.

B. Current Integration Mehtod

The current integration (CI) method is a real-time battery SOC estimation method based on CI. The SOC at time t is shown in (1), and the basic principle of the CI method is to add all the charges flowing into and out of the battery in terms of ampere-hours [11]. The CI method requires determining the initial SOC and performing accurate current measurement. Although the initial SOC can be estimated via the CI method using the OCV-SOC data sheet, the OCV-SOC data sheet becomes inaccurate as the battery ages. Thus, It is needed to renew the data sheet to perform accurate battery SOC estimation. In addition, there is the disadvantage of accumulated errors over time.

$$SOC(t) = SOC(t_0) + \int_{t0}^{t} \frac{I(t)dt}{C_N} \quad (1)$$

Where, t_0 is the initial time, $I(t)$ is the battery current, and C_N is the nominal capacity of the battery.

III. Proposed SOC Estimation Methods based on Neural Network

In this paper, we propose methods for estimating the lead-acid battery SOC using an FFNN, an RNN, and an ANFIS. To perform lead-acid battery SOC estimation through the neural networks, we selected the voltage and the current as input parameters for these models. The reason for using these two parameters is that they can be measured easily, and have important relationships with the battery SOC. A schematic diagram of the lead-acid battery SOC estimation methods based on the proposed neural networks is shown in Fig. 1. Also, SOC at time t is defined as shown in (2) by us, to set the battery SOC range as 0% to 100% [12].

$$SOC(t) = SOC(t_0) + \int_{t0}^{t} \frac{I(t)dt}{C_M} \quad (2)$$

Where, t_0 is the initial time, $I(t)$ is the battery current, and C_M is the measured capacity of the battery.

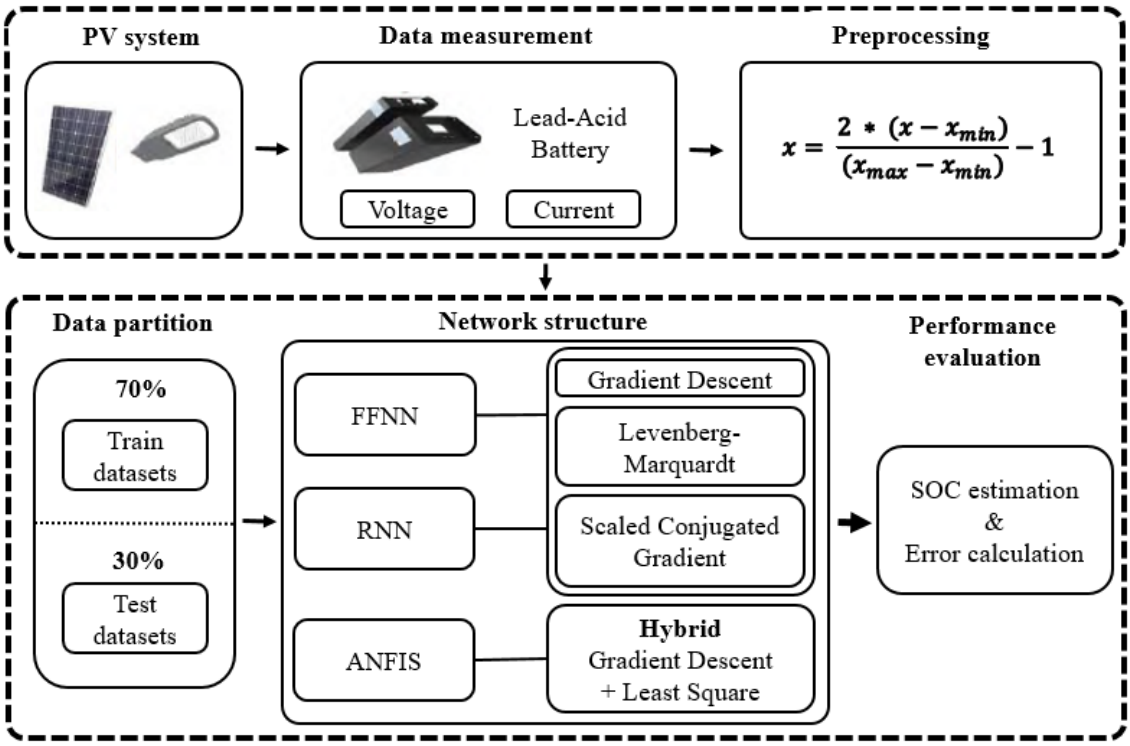

Fig. 1. Schematic Diagram of the Neural Network-based Lead-Acid Battery SOC Estimation.

A. Feed-Forward Neural Network

The feed-forward neural network (FFNN) is a multilayer neural network with one or more hidden layers. The basic structure is composed of an input layer, a hidden layer, and an output layer, as shown in Fig. 2. Each layer is composed of nodes (neurons) and the weights connected between each neuron.

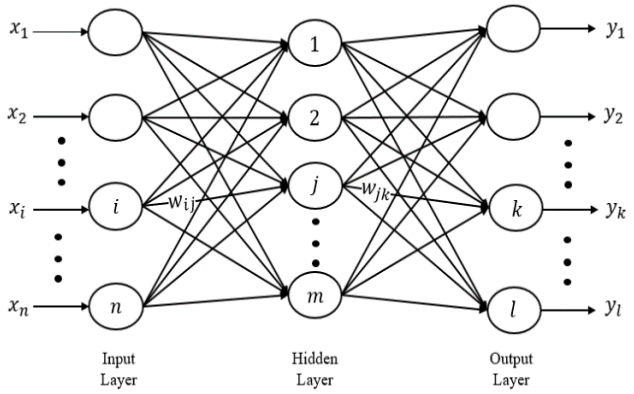

Fig. 2. Structure of the FFNN.

The input pattern is multiplied by the weights between the input layer and the hidden layer, and it passes through the activation function of the hidden layer. The output of node j in the hidden layer at epoch p and output of node k in the output layer at epoch p, as given by (3) and (4).

$$y_j(p) = f_j(\sum_{i=1}^{n} x_i(p) w_{ij}(p)) \quad (3)$$

Where, n is the number of input samples, and f_j is the activation function of the hidden node j.

$$y_k(p) = f_k(\sum_{j=1}^{m} y_j(p) w_{jk}(p))) \quad (4)$$

Where, m is the number of hidden nodes, f_k is the activation function of the output node k.

The error of the output pattern for the target pattern at output node k at epoch p is defined by (5). The error is calculated and the weight is modified again through the backpropagation method. As this process repeats, it is corrected to the optimal weight to obtain the target output pattern using the input pattern. If this model is generalized well, we can obtain the target pattern for un-learned input patterns.

$$e_k(p) = d_k(p) - y_k(p) \quad (5)$$

Where, $d_k(p)$ is the target value, and $y_k(p)$ is the output value at output node k at epoch p.

B. Recurrent Neural Network

The recurrent neural network (RNN) is a representative model for processing sequence data. It differs from the generalization model of the existing FFNN structure, as shown Fig. 3. The most important feature of an RNN is that it has the state of the hidden layer at time t, as shown in (6).

$$h(t) = f_h(x(t)W_{ih} + h(t-1)W_{hh}) \quad (6)$$

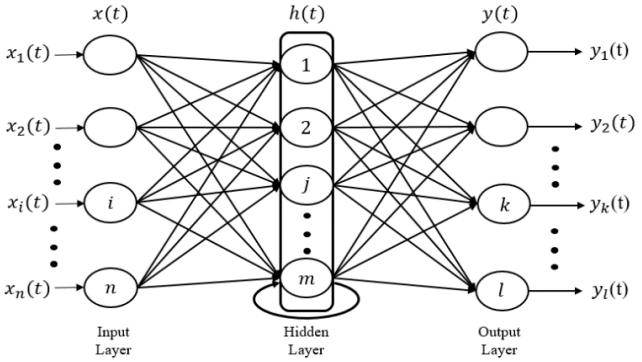

Fig. 3. Structure of the RNN.

Where, $h(t-1)$ represents the activation values of the hidden layer at time $t-1$, which are replicated at time t and accumulate past information, f_h is the activation function of the hidden layer, $x(t)$ is the input at time t, W_{ih} is the weight matrix between $x(t)$ and $h(t)$, and W_{hh} is the connection matrix between $h(t-1)$ and $h(t)$. The output at time t as shown in (7).

$$y(t) = f_o(h(t)W_{ho}) \quad (7)$$

Where, f_o is the activation function of the output layer, and W_{ho} is the weight matrix between $h(t)$ and $y(t)$.

C. Adaptive Neuro Fuzzy Inference System

In the adaptive nuero fuzzy inference system (ANFIS) model [13], the sugeno fuzzy model is a systematic method for generating fuzzy rules from the input data and the output data sets. It uses a hybrid learning method, which combines the least-squares estimator and the GD method. The hybrid learning method has the advantage of converging quickly [14].

As shown in Fig. 4, the ANFIS is divided into six layers: the input layer, fuzzification layer, rule layer, normalization layer, defuzzification layer, and summation neuron layer. Term x_n represents the input values, A_n and B_n are fuzzy sets, f_n is the output function, and y is the output at epoch p, as shown in (8).

$$y(p) = k_0(p) + k_1(p)x_1(p) + \cdots + k_m(p)x_m(p) \quad (8)$$

Where, k_{i0}, k_{i1}, and k_{i2} are the sets of arguments for rule i.

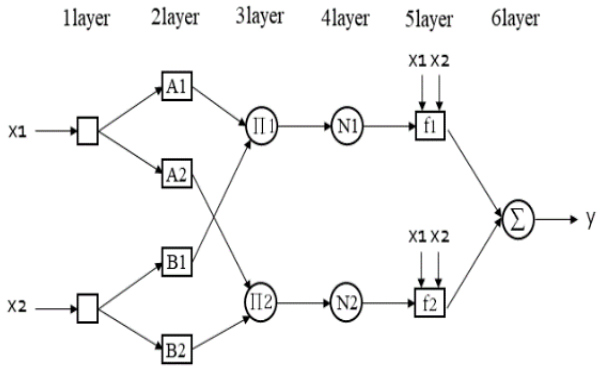

Fig. 4. Structure of the ANFIS.

In the ANFIS learning algorithm, each epoch is composed of a forward calculation and a backward calculation. In the forward calculation, the training set of the input pattern is input to the ANFIS, the neuron output is calculated for each layer, and the rule consequent factor is obtained via least-squares estimation.

1) GD training method: The gradient descent (GD) method minimizes the error between the target output and the output value by adjusting weights. The error gradient is transmitted in the reverse direction at the k[th] output node at epoch p and is expressed as (9)

$$\delta_k(p) = \dot{f}_k(p) \cdot e_k(p) \tag{9}$$

Where, $e_k(p) = d_k(p) - y_k(p)$, and $\dot{f}_k(p)$ is the derivative of the activation function of output node k at epoch p.

The slope and the weight (w_{kj}) of the cost function for the weight between the hidden layer and the output layer are improved at the $p + 1$[th] epoch to (11).

$$\Delta w_{jk}(p) = -\eta \cdot y_j(p) \cdot \delta_k(p) \tag{10}$$

$$w_{jk}(p+1) = w_{kj}(p) + \Delta w_{kj}(p) \tag{11}$$

Where, $\Delta w_{jk}(p)$ is the weight improvement amount between the j[th] hidden node and the k[th] output node at epoch p, and η is the learning rate.

Similarly, the error signal at the j[th] hidden node and the adjustment of the weight (w_{ij}) between the input layer and the hidden layer at the $p + 1$[th] epoch is as follows (12),(13), and (14).

$$\delta_j(p) = \dot{f}_j(p) \cdot \sum_{k=1}^{N} \delta_k(p) \cdot w_{jk}(p) \tag{12}$$

$$\Delta w_{ij}(p) = -\eta \cdot x_i(p) \cdot \delta_j(p \tag{13}$$

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p) \tag{14}$$

Where, $\Delta w_{ij}(p)$ is the weight improvement amount between the i[th] input node and the j[th] hidden node, and $\delta_j(p)$ is the error gradient that is reversed from the hidden layer to the input layer at epoch p.

2) SCG training method: The scaled conjugate gradient (SCG) training method is known to be effective for large problems. It uses the second-order information, without the line-search process. Thus, amount of using memory can be reduced by reducing the amount of computation of gradient information. The final SCG algorithm is detailed below [15].

① Choose the weight vector $\tilde{w}_1$ and scalars $0 < \sigma \le 10^{-4}$,

$0 < \lambda_1 \le 10^{-6}, \bar{\lambda}_1 = 0$.

Set $\tilde{p}_1 = \tilde{r}_1 = -E'(\tilde{w}_1), p = 1,$ and $success = true$.

② If *success* = *true*, calculate the second-order information:

$$\sigma_p = \sigma / |\tilde{p}_p|,$$

$$\tilde{s}_p = (E'(\tilde{w}_p + \sigma_p \tilde{p}_p) - E'(\tilde{w}_p))/\sigma_p,$$

$$\delta_p = \tilde{p}_p^T \tilde{s}_p.$$

③ Scale δ_p: $\delta_p = \delta_p + (\lambda_p - \bar{\lambda}_p)|\tilde{p}_p|^2$.

④ If $\delta_p \le 0$, make the Hessian matrix positive definite:

$$\bar{\lambda}_p = 2(\lambda_p - \delta_p / |\tilde{p}_p|^2),$$

$$\delta_p = -\delta_p + \lambda_p |\tilde{p}_p|^2,$$

$$\lambda_p = \bar{\lambda}_p.$$

⑤ Calculate the step size:

$$\mu_p = \tilde{p}_p^T \tilde{r}_p,$$

$$\alpha_p = \mu_p / \delta_p.$$

⑥ Calculate the comparison parameter:

$$\Delta_p = 2\delta_p [E(\tilde{w}_p) - E(\tilde{w}_p + \alpha_p \tilde{p}_p)] / \tilde{\mu}_p^2.$$

⑦ If $\Delta_k \ge 0$, a successful reduction in the error can be made:

$$\tilde{w}_{p+1} = \tilde{w}_p + \alpha_p \tilde{p}_p,$$

$$\tilde{r}_{p+1} = -E'(\tilde{w}_{p+1}),$$

$$\bar{\lambda}_p = 0, \textit{success} = \textit{true}.$$

If k mod $N = 0$, restart the algorithm:

$$\tilde{p}_{p+1} = \tilde{r}_{p+1}$$

else:

$$\beta_p = (|\tilde{r}_{p+1}|^2 - \tilde{r}_{p+1}^T \tilde{r}_p)/\mu_p,$$

$$\tilde{p}_{p+1} = \tilde{r}_{p+1} + \beta_p \tilde{p}_p.$$

If $\Delta_p \ge 0.75$, reduce the scale parameter:

$$\lambda_p = \tfrac{1}{4} \lambda_p.$$

else:

$$\bar{\lambda}_p = \lambda_p,$$

success = *false*.

⑧ If $\Delta_p < 0.25$, increase the scale parameter:

$$\lambda_p = \lambda_p + (\delta_p (1 - \Delta_p) / |\tilde{p}_p|^2).$$

⑨ If the steepest descent direction $\tilde{r}_p \ne \tilde{0}$, set $p = p + 1$ and go to ②; else, terminate and return $\tilde{w}_{p+1}$ as the desired minimum.

3) LM training method: The levenberg-marquardt (LM) training method [16] is a deformation of the Newton method. The algorithm has a faster convergence of second orders, fewer iterations, and does not need to compute the Hessian matrix. For some network models with few parameters, the training speed of the algorithm is higher [17]. Let N be the vector of the weight and bias of each layer in the iterative training is given by (15). The delta N is the adjustment

quantity of N, and adjusting N means the regulation of the weights and thresholds of each layer in the network; finally, the goal of training the network is achieved.

$$\mathrm{N} = \begin{bmatrix} w_{11} & w_{12\ldots}w_{IH} & \theta J\,(1)\ldots\theta J(H) \\ V_{11} & V_{12\ldots}V_{HO} & \theta I\,(1)\ldots\theta I(O) \end{bmatrix} \quad (15)$$

$$\mathrm{P(N)} = \sum_{i=1}^{I} \mathrm{e}_i(N)^2 \quad (16)$$

Where, P(N) is expression function, $\mathrm{e}_i(N)^2, \mathrm{i} = 1 \rightarrow$ I represents the square of error.

$$\nabla \mathrm{P(N)} = J^T(\mathrm{N})\mathrm{E(N)} \quad (17)$$

$$\nabla^2 \mathrm{P(N)} = J^T(\mathrm{N})\mathrm{E(N)} + \mathrm{S(N)} \quad (18)$$

Where, E(N) is expressed as $\mathrm{e}_i(\mathrm{N})$, $\mathrm{i} = 1 \rightarrow$ I constituent vectors, J(N) is the Jacobian matrix given by (19), and S(N) is the error function given by (20).

$$J(\mathrm{N}) = \begin{bmatrix} \frac{\partial \mathrm{e}_1(\mathrm{N})}{\partial \mathrm{N}_1} & \frac{\partial \mathrm{e}_1(\mathrm{N})}{\partial \mathrm{N}_2} & \frac{\partial \mathrm{e}_1(\mathrm{N})}{\partial \mathrm{N}_\mathrm{i}} \\ \frac{\partial \mathrm{e}_2(\mathrm{N})}{\partial \mathrm{N}_1} & \frac{\partial \mathrm{e}_2(\mathrm{N})}{\partial \mathrm{N}_2} & \frac{\partial \mathrm{e}_2(\mathrm{N})}{\partial \mathrm{N}_\mathrm{i}} \\ \frac{\partial \mathrm{e}_\mathrm{I}(\mathrm{N})}{\partial \mathrm{N}_1} & \frac{\partial \mathrm{e}_\mathrm{I}(\mathrm{N})}{\partial \mathrm{N}_2} & \frac{\partial \mathrm{e}_\mathrm{I}(\mathrm{N})}{\partial \mathrm{N}_\mathrm{i}} \end{bmatrix} \quad (19)$$

$$\mathrm{S(N)} = \sum_{i=1}^{I} \mathrm{e}_i\,(N)\nabla^2 \mathrm{e}_i(N) \quad (20)$$

Because the LM algorithm is an improved form of the Gauss Newton method, ΔN is given as shown in (21).

$$\Delta \mathrm{N} = \mathrm{J}^T(N)E(N)[J^T(N)J(N) + \mu I]^{-1} \quad (21)$$

Where, I is the unit matrix, and $\mu > 0$ is a constant.

When μ = 0 is used for the Gauss Newton method, when larger, LM approaches the small-step GD method. For training, the modification factor is changed from μ to α. If the training fails, μ is increase or, decrease it. The solution of $\mathrm{J}^T(N)E(N)[J^T(N)J(N) + \mu I]^{-1}$always exists, because of it is positive value. Thus, the LM algorithm is superior to the Gauss Newton method. The LM algorithm steps are as follows.

① The allowable values, coefficients, thresholds, weights of error training, and k = 0 (k is the number of iterations) are initialized.

② Calculate the output of network and the expression function P(N) and Jacobian matrix J(N).

③ Calculate ΔN.

④ If P(N) < ε, the end; otherwise, use $N + \Delta N$ as the weights and thresholds to recalculate the expression function P(N). When P(N) is less than the P(N) of ②, set $\mu = \mu/\alpha$ and $p = p + 1$, and return to ②; otherwise, set $\mu = \mu \cdot \alpha$, and return to ③.

IV. Experiment Results

A. Data-Acquisition Environment

We set up the experimental environment to obtain the real data required for lead-acid battery SOC estimation, as shown in Fig. 5.

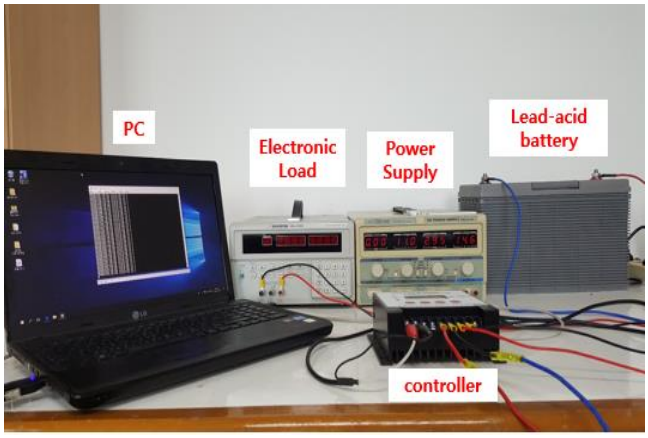

Fig. 5. Experimental Setup.

TABLE I. Specifications of Experimental Equipment

Item	Specification	Quantity
Power supply	Model: UP-150DT Max values: 5 A/30 V (DUAL)	1EA
Lead-acid battery	Model: KB100-12 Nominal capacity: 12 V, 100 Ah	1EA
Electronic load	Model: PEL-300 Power: 1–300 W C.V mode: 3–6 V C.C mode: 6 mA–60 A	1EA

The experimental environment composed of a lead-acid battery, an electronic load, a power supply, a battery controller, and a PC. Further, by connecting the battery controller and the PC in series, the real-time battery data are monitored and collected by the PC. The power supply performs battery charging, and the electronic load performs battery discharging. The battery controller includes a protection function for over-charge or over-discharge of the battery and monitoring of the current and voltage of the battery in real time. The detailed specifications of the experimental equipment are shown in TABLE I.

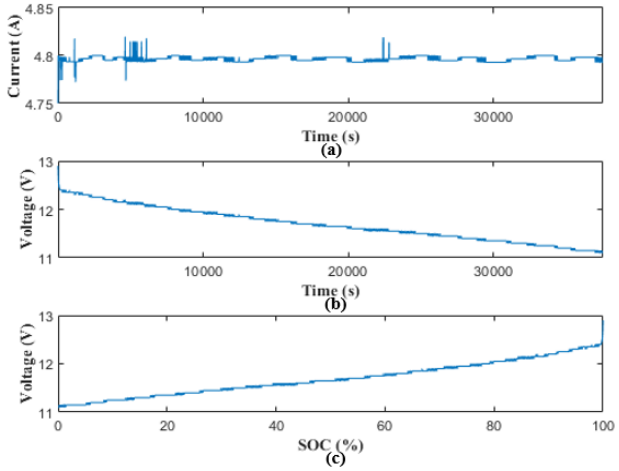

Fig. 6. One Cycle of Discharge: (A) Current, (B) Voltage, and (c) Voltage Against the SOC.

In addition, the collected data are nominalized to a range of between -1 and 1 for the positive effect that the network model has faster convergence in training, as shown in (22).

$$x = \frac{2(x - x_{\min})}{x_{\max} + x_{\min}} - 1 \quad (22)$$

Where, $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of x from the data that we collected.

A solar controller was used to prevent over-charging and over-discharging when the data were acquired. The battery was discharged at a constant load from the fully charged state to the discharge end voltage. The battery voltage range was 11.1 ~ 12.9. A graph of the battery discharge is shown in Fig. 6.

B. Configuration of Network

The neural network models are implemented and tested by using MATLAB. The configuration of the FFNN and RNN model is as follows: the input nodes are 2, the output node is 1, the hidden nodes are 30, the learning rate is 0.01, and the training error goal is 0.0005. We used the hyperbolic tangent sigmoid function as the activation function of the hidden layer, and the linear transfer function as the activation function of the output layer. The initial weights were random values. The cost function was the mean squared error (MSE) and is given by (23).

$$MSE = \frac{1}{n}\sum_{p=1}^{n}\sum_{k=1}^{l}(d_k(p) - y_k(p))^2 \quad (23)$$

Where, n is the number of training sets, $d_k(p)$ is the k^{th} target value, and $y_k(p)$ is the output at output node k at epoch p.

The configuration of the ANFIS model is as follows: the input nodes are 2, the output node is 1, the membership functions are Gaussian, and the number of rule neurons is 25. Training is performed until the root-mean-square error (RMSE) reaches the error tolerance 0.01, and the RMSE is given by (24).

$$RMSE = \sqrt{\frac{1}{n}\sum_{p=1}^{n}(d(p) - y(p))^2} \quad (24)$$

Where, n is the number of training sets, and $d(p)$ and $y(p)$ are the target value and output value at epoch p.

C. SOC Estimation Results

We used the data obtained from the experimental setup presented in Section 4.1 and compared the proposed methods for estimation of the lead-acid battery SOC. First, the FFNN model was tested by applying three training modes, and the SOC estimation results are shown in Fig. 7, Fig. 8, and Fig. 9.

The FFNN-GD method passed 100,000 epochs to converge to the desired MSE during training, and the test MSE was 2.11. FFNN-SCG recorded 1,401 epochs, and the MSE was 2.03, which was better than that of FFNN-GD. FFNN-LM converged on 35 epochs, and the MSE was 1.78 as the test data. Thus, the MSE and epochs were superior to those of the other two training methods.

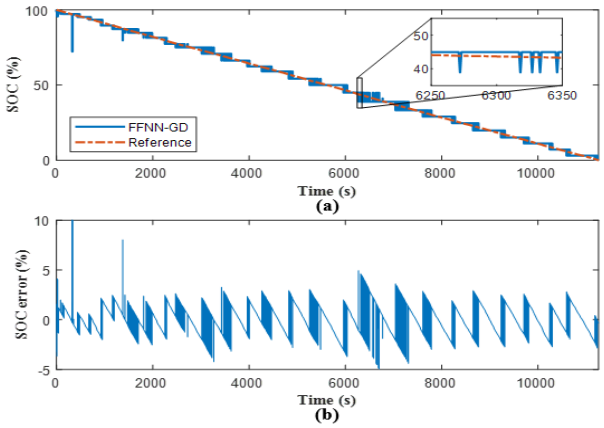

Fig. 7. SOC Estimation using FFNN-GD: (a) SOC and (b) SOC Error.

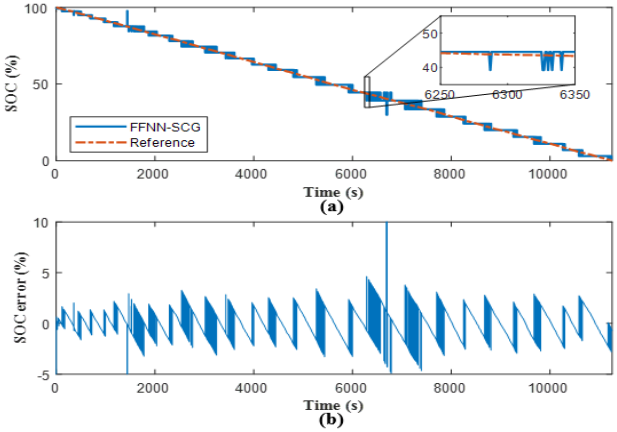

Fig. 8. SOC Estimation using FFNN-SCG: (a) SOC and (b) SOC Error.

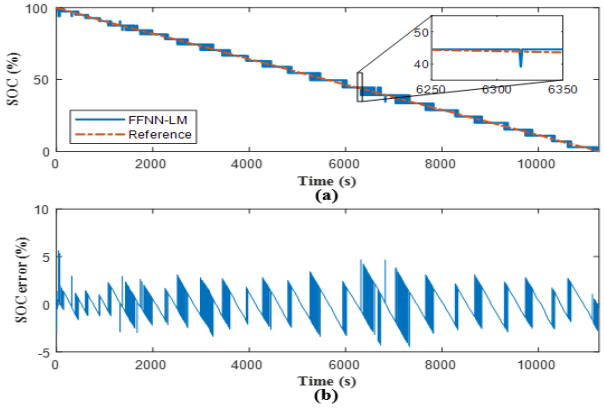

Fig. 9. SOC Estimation using FFNN-LM: (a) SOC and (b) SOC Error.

Second, the RNN model was tested by applying the same three training modes, and the SOC estimation results are shown in Fig. 10, Fig. 11, and Fig. 12. The RNN-GD method passed 100,000 epochs to converge to the desired MSE during training, and the test MSE was 1.67. The RNN-SCG recorded 764 epochs, and the MSE was 1.19, which was better than that of the RNN-GD method. RNN-LM converged to epochs. The MSE of 1.08 was better than those of RNN-GD and RNN-SCG.

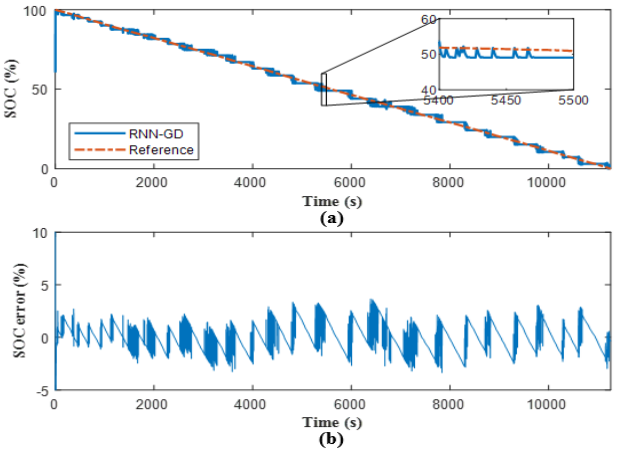

Fig. 10. SOC Estimation using RNN-GD: (a) SOC and (b) SOC Error.

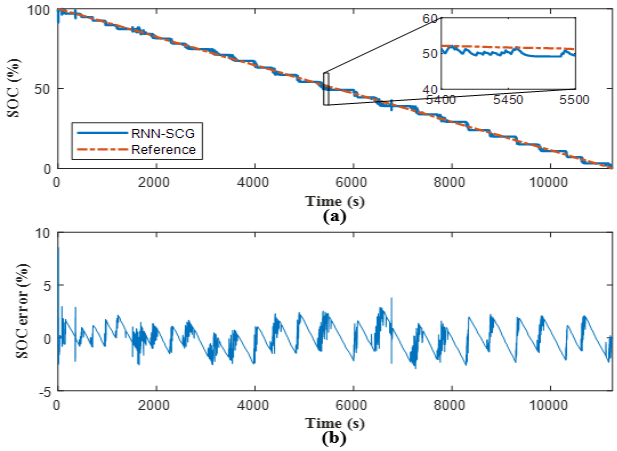

Fig. 11. SOC Estimation using RNN-SCG: (a) SOC and (b) SOC Error.

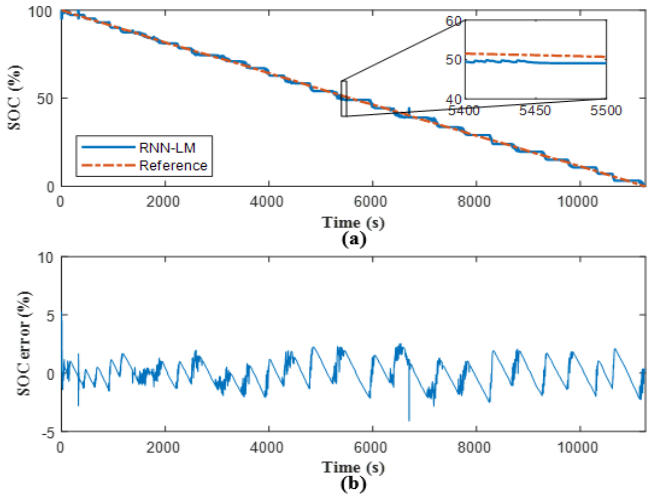

Fig. 12. SOC Estimation using RNN-LM: (a) SOC and (b) SOC Error.

Finally, SOC estimation was performed with ANFIS using a hybrid training method, and the results are shown in Fig. 13. The training was completed in only one epoch, and the MSE was 1.82 as a result of the SOC estimation with test samples. All the experimental results are comprehensively compared in Fig. 14 and TABLE II.

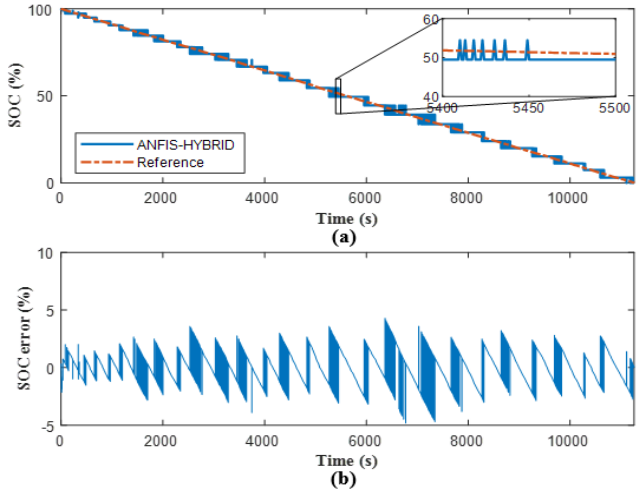

Fig. 13. SOC Estimation using ANFIS-Hybrid: (a) SOC and (b) SOC Error.

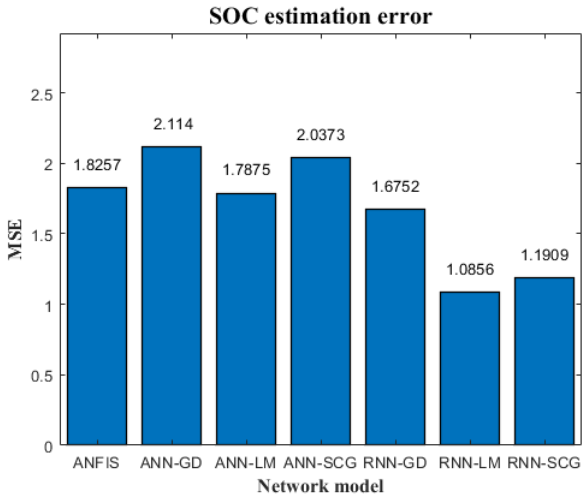

Fig. 14. Comparison of the SOC Estimation Error.

TABLE II. COMPARISON OF TRAINING EPOCHS AND MSE

Para-meter	Network model						
	FFNN			*RNN*			*ANFIS*
	GD	*SCG*	*LM*	*GD*	*SCG*	*LM*	*Hybrid*
Epochs	Over 100, 000	1,401	35	Over 100, 000	767	6	1
MSE	2.11	2.03	1.78	1.67	1.19	1.08	1.82

V. EXPERIMENT RESULTS

Three intelligent models (FFNN, RNN, and ANFIS) for estimating the lead-acid battery SOC in PV systems were presented. Additionally, we compared the proposed methods with regard to the MSE and epochs.

The experimental results are as follows. For the FFNN and RNN, we used three training methods and found that FFNN-LM (MSE of 1.78, 35 epochs) and RNN-LM (MSE of 1.08, 6 epochs) demonstrated excellent performance. The ANFIS

used the hybrid learning method with GD, and according to the least-squares method, the MSE was 1.82, with 1 epoch. Moreover, the convergence speed was the highest among all the models. In summary, RNN-LM can learn at a high speed and is the most accurate among all the methods; thus, the RNN-LM method is suitable for estimating the lead-acid battery SOC because it can be generalized to the greatest extent. In a future study, the proposed methods will be applied to a lithium battery.

REFERENCES

[1] C. K. Park, "Study on the obsolescence forecasting judgement of PV systems adapted micro-inverters", Journal of Korea Multimedia Society, vol. 18, no. 7, pp. 864-872, 2015.

[2] J. H. Jung, J. H. Jung, "New State-of-Charge Polynomial using Hermite Interpolation", Journal of the Institute of Electronics Engineers of Korea, vol.48, no.1, pp. 9 – 17, 2011.

[3] M. Coleman, C. K. Lee, C. Zhu, and W. G Hurley, "State-of-Charge Determination From EMF Voltage Estimation: Using Impedance, Terminal Voltage, and Current for Lead-Acid and Lithium-Ion Batteries", IEEE transactions on industrial electronics, vol. 54, no. 5, Oct, 2007.

[4] K. S. Ng, C.-S. Moo, Y.-P. Chen, and Y.-C. Hsieh, ''Enhanced coulomb counting method for estimating state-of-charge and state-of-health of lithium-ion batteries,'' Appl. Energy, vol. 86, no. 9, pp. 1506–1511, 2009.

[5] SchurmFFNN J, "Pattern classification, a unified view of statistical and neural approaches", john Wiley and Sons, New York, 1996.

[6] Kohonen T, Self-organizing maps, Springer, Berlin, 1997.

[7] Fausett L, Fundamental of neural netwroks, Prentice Hall, 1994.

[8] Timmermans, A.J.M and A.A hulzebosch, "Computer vision system for on-line sorting of pot plants using an artificila neural network classifier,"Computer and Electronics in Agriculture, vol. 15, no. 1, pp. 41-55, May,1996.

[9] Th.F. Elshatter, M.T. Elhagry, E.M. Abou-Elzahab, A.A.T. Elkousy,"Fuzzy modeling of photovoltaic panel equivalent circuit", in: Proceedings of the Photovoltaic Specialists Conference, vol. 15, no. 22,pp. 1656–1659, Sep. 2000.

[10] M. AbdulHadi, A.M. Al-Ibrahim, G.S. Virk, "Neuro-fuzzy-based solar panel model", IEEE Trans. Energy Convers. vol. 19, no. 3, pp. 619–624,Aug. 2004.

[11] M. S. H. Lipu, M. A. Hannan, A. Hussain, and M. H. M. Saad, "Optimal BP neural network algorithm for state of charge estimation of lithium-ion battery using PSO with PCA feature selection," J. Renew. Sustain. Energy, vol. 9, no. 6, p. 64102, Nov. 2017.

[12] K. S. Yu, B. K. Kim, D. J. Kim, M. S. Jang, H. S. Go and H. C. Kim, "A State-of-Charge estimation using extended Kalman filter for battery of electric vehicle" Journal of Korea Academia-Industrial cooperation Society, vol. 18, no. 10, pp. 15-23, 2017.

[13] Michael Negnevitsky, Artificial Intelligence, 2rd ed., Addison Wesley.

[14] S. H. Par, K. J. Lim, S. H. Kang, J. M. Seo, and Y. G. Kim, "Comparison of recognition rates between BP and ANFIS with FCM clustering method on off-line PD diagnosis of defect models of traction motor stator coil," KIEE International Transactions on Electrophysics and Applications, vol. 5-C, no. 4, pp. 138-142, 2005.

[15] M. F. Moller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", Neural Networks, Vol. 6, pp. 525-533, 1993.

[16] Zeng Min, Liang Xiao, Lin Cao, Hangcheng Yan, "Application of the Neural Network in Diagnosis of Breast Cancer Based on Levenberg-Marquardt Algorithm", International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), p.268 – 272, 2017.

[17] Arun. D. Kulkarni. Fuzzy Neural Network Models for Classification. Applied Intelligence, 2000, 1 2(3): 207-215.