Received October 29, 2020, accepted November 22, 2020, date of publication November 27, 2020,
date of current version December 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3041007

Real-Time Optimal Power Flow Using Twin Delayed Deep Deterministic Policy Gradient Algorithm

JONG HA WOO[1], LEI WU[2], (Senior Member, IEEE), JONG-BAE PARK[1], (Member, IEEE), AND JAE HYUNG ROH[1], (Member, IEEE)

[1]Department of Electrical and Electronics Engineering, Konkuk University, Seoul 05029, South Korea
[2]Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA

Corresponding author: Jae Hyung Roh (jhroh@konkuk.ac.kr)

This work was supported in part by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry, and Energy (MOTIE), Republic of Korea, under Grant 20194310100060, and in part by the Human Resources Program in Energy Technology of KETEP and MOTIE under Grant 20204010600220.

ABSTRACT The general concept of AC Optimal Power Flow (ACOPF) refers to the economic dispatch planning under electric network constraints. Moreover, each instance with the entire network must be solved in real-time (i.e., every five minutes) to ensure cost-effective power system operation while satisfying power balance equation. As the operation of power systems penetrated with intermittent renewable energy becomes more complicated, this article proposes Deep Neural Network (DNN) and Levenberg-Marquardt backpropagation-based Twin Delayed Deep Deterministic Policy Gradient (TD3) approach to improve computational performance of ACOPF. Specifically, because the ACOPF model shall consider prevailing constraints of the power system, including power balance equation, we set the appropriate reward vector in the training process to build our own policy. Furthermore, we add random Gaussian noise to individual net loads for representing uncertainty characteristics introduced by renewable energy sources. Finally, the proposed model is compared with the MAT-POWER solution on the IEEE 118-bus system to demonstrate its efficacy and robustness.

INDEX TERMS Deep deterministic policy gradient, deep reinforcement learning, Levenberg Marquardt, optimal power flow, twin delayed deep deterministic policy gradient.

I. INTRODUCTION

In a deregulated power system, ACOPF is the primary tool to offer the power system operation solution economically with high quality, which is a large-scale, multi-dimensional, non-convex, non-linear, and constrained optimization problem [1]. It is difficult to determine the generator's economic outputs due to the non-linear cost functions of generators and the rapid changes in load. Especially, as renewable energy resources are increasingly integrated into the power system, the random and intermittent characteristics of renewable energy induce significant challenges in terms of how to control and dispatch these resources in real-time [2]. Moreover, the uncertainty of renewable energy output needs to be considered in OPF for optimum generator dispatch considering ramp-rate, increasing renewable energy integration proportion, where the forecasting error is inevitable even though advanced prediction techniques are utilized [3].

The Economic Dispatch (ED) addresses these issues to a certain extent but does not consider the loss, reactive power, and transmission line congestion [4]. To this end, Optimal Power Flow (OPF) problems are able to address these economic dispatch problems more realistically and reasonably. Since then, there has been extensive research on OPF algorithms, thanks to the recent development of optimization techniques and computational technologies [5].

DC OPF has the advantage of computational efficiency, but it has the disadvantage of not being able to achieve precise results because of DC approximations. In comparison, AC OPF allows for the consideration of both active and reactive power flows and terminal voltages of all generators. Although AC OPF is the most common form of OPF and

The associate editor coordinating the review of this manuscript and approving it for publication was Jason Gu.

can produce precise results, it has the disadvantage of slow computation.

As such, it is a challenging task to obtain general real-time ACOPF solutions from existing algorithms. Therefore, ACOPF solutions with DNN (Deep Neural Network) and DRL (Deep Reinforcement Learning) were introduced to speed up the calculation process [6], [7], [8]. It has been noted that DNN and DRL approaches can solve the ACOPF problem of complex, larger dimensions, and continuous state space. Existing research [7] adopted DRL to solve AC OPF problem while considering ramp-rate, total cost, and load changes in large scale systems.

This article aims to determine the economic operations of generators when loads in the power system present a random pattern. We test the proposed model on the IEEE 118-bus system and compare the results with MAT-POWER MIPS (MAT-POWER Interior Point Solver), an OPF problem solver, to verify effectiveness and efficiency of the proposed approach. To evaluate the generator's ramp-rate constraints, we vary the net load from 0.8 to 1.2 to simulate realistic load changes. In addition, random Gaussian noise is added to each load to simulate load uncertainties. Therefore, these tests will illustrate the computational efficiency and robustness of the proposed approach against load disturbances.

Power flow calculation intends to find out nodal voltage profiles of the power system with given nodal power supplies and demands. Once nodal voltages are calculated, power flows in transmission lines and generators' power outputs can be easily calculated. In addition, it is essential to consider network loss when power exchanges among buses. In this case, because the number of equations is less than that of variables, specifying all the outputs of generators cannot directly solve this problem. Thus, one generator is usually defined as the slack generator to take over all system losses and guarantees the equation's rank [9]. It means the slack generator is automatically designated by the power flow calculation to determine outputs (active power) of the rest of the generators. To this end, we build a machine learning model to consider total loss and satisfy the power balance equation effectively.

In Section II, we talk about deep learning and the ACOPF problem. Section III describes the proposed approach. Section IV conducts studied case to verify effectiveness of the proposed model as compared to MAT-POWER. It confirms that the slack generator's output from the proposed model is the correct answer considering the other generator's output. This article is concluded in Section V.

II. ACOPF PROBLEM AND DEEP LEARNING

A. ACOPF PROBLEM

The objective function of OPF in Eq. (1) is expressed by minimizing the summation of the generator's production costs, while satisfying prevailing constraints.

$$\min \sum_{\mathrm{i=1}}^{\mathrm{Ng}} \mathrm{C_{gi}}(P_{Gi}^t) \tag{1}$$

$$\text{subject to } V_i^{t\,\min} \leq V_i^t \leq V_i^{t\,\max} \tag{2}$$

$$\theta_i^t \leq \theta_i^t \leq \theta_i^{t\,\max} \tag{3}$$

$$P_{Gi}^{t\ \ \min} \leq P_{Gi}^t \leq P_{Gi}^{t\ \ \max} \tag{4}$$

$$Q_{Gi}^{t\ \ \min} \leq Q_{Gi}^t \leq Q_{Gi}^{t\ \ \max} \tag{5}$$

$$|F_j^t| \leq \mathrm{F_j^{max}} \tag{6}$$

$$\sum_{\mathrm{i=1}}^{\mathrm{Ng}} S_{Gi}^t - \sum_{\mathrm{i=1}}^{\mathrm{Ng}} S_{Li}^t = S_{loss}^t \tag{7}$$

$$\mathrm{P_G^{t-1} - Ramp_{down} < P_G^t < P_G^{t-1} + Ramp_{up}} \tag{8}$$

$\mathrm{N_g}$ is the total number of generators in the system. The subscript i represents the bus index, $\mathrm{P_G}$ represents the generator's output, and C represents the generator's cost function. V and θ refer to magnitude and angle of voltages in each bus, and their limitations are described as in Eq. (2) and (3). Since each generator has output limitation, Eq. (4) and (5) show maximum and minimum values for the operating range. The left side of the Eq. (6) represents the power flow of the jth transmission line F_j^t, and the right side represents the maximum capacity $\mathrm{F_{j\,max}}$ of the line. Eq. (7) means that the summation of the total apparent power generation S_G^t and the total system loss S_{loss}^t under the condition of the energy balance is consistent with the demand S_L^t. Ramp-rate constraint (11) considers ramp limitation between the current step P_G^t and the previous step P_G^{t-1}.

MAT-POWER provides a function to solve OPF problems. This function, named MAT-POWER Interior Point Solver (MIPS), is a MATLAB language M-files package for solving non-linear programming problems (NLPs), using a primal-dual interior-point method.

B. DEEP NEURAL NETWORK

A standard neural network (NN) consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Input neurons get activated through sensors perceiving the environment; other neurons get activated through weighted connections from previously active neurons via backpropagation [10].

A backpropagation algorithm with chain rules can be used to optimally update parameters of the multi-layer feed-forward neural network. We define the target (loss) function for solving optimization problems. In general, the squared (Euclidean) loss can be used as loss function. If we define the d-dimensional target output as $t = [t1, \ldots, td]$ and the estimated output as $x = [x1, \ldots, xd]$, the summation of squared loss can be defined as shown in Eq. (9):

$$\mathrm{E} = \sum_{\mathrm{i=1}}^{\mathrm{d}} (\mathrm{x_i} - \mathrm{t_i})^2 \tag{9}$$

To optimize the mentioned loss function, DNN (Deep Neural Network) uses the method of backpropagation, which includes Gradient descent, Gauss-Newton, and Levenberg Marquette, and so on. Eq. (10) represent how to update parameters in different methods.

Gradient descent

$$p_{k+1} = p_k - 2\lambda_k J_r^T(p_k) r(p_k), \quad k \geq 0$$

Gauss-Newton

$$p_{k+1} = p_k - (J_r^T J_r)^{-1} J_r^T r(p_k), \quad k \geq 0 \tag{10}$$

Levenberg-Marquardt

$$p_{k+1} = p_k - (J_r^T J_r + \mu_k \mathrm{diag}(J_r^T J_r))^{-1} J_r^T r(p_k), \quad k \geq 0$$

where,

$$J_r(p) = \begin{pmatrix} \frac{\partial r_1(p)}{\partial p_1} & \cdots & \frac{\partial r_1(p)}{\partial p_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_n(p)}{\partial p_1} & \cdots & \frac{\partial r_n(p)}{\partial p_m} \end{pmatrix}$$

$$r(p) = \begin{pmatrix} r_1(p) \\ r_2(p) \\ \vdots \\ r_n(p) \end{pmatrix} = \begin{pmatrix} y_1 - f(x_1, p) \\ y_2 - f(x_2, p) \\ \vdots \\ y_n - f(x_n, p) \end{pmatrix}$$

The parameters of the model are p_k, the updated parameters in next step are p_{k+1}, errors(residual) between observed value and actual value are $r(p)$, and J_r is briefly expressed in the Jacobian matrix $J_r(p_k)$ of vector $r(p)$ in the k^{th} step.

The Gradient Descent is a method to find the smallest point that minimizes the error function, by moving the gradient in the opposite direction via step size proportional to the size of the gradient [11]. On the other hand, Gauss-Newton is a method to find the solution by considering the gradient and curvature of the function together. Thus, the Gauss-Newton method has the advantage of finding the solution much more accurately and quickly than the gradient descent method. However, the expression requires the calculation of the inverse of the curvature $J_r^T J_r$, which represents the approximation matrix of Hessian [12]. If this matrix is close to the singular matrix, the calculated inverse matrix is numerically unstable, leading to the divergence.

In comparison, the Levenberg-Marquardt is a way to mitigate the risk of divergence and to find more stable roots by adding $\mu \times \mathrm{diag}(J_r^T J_r)$ to $J_r^T J_r$ [13]. The diagonal elements of $J_r^T J_r$, an approximate matrix for Hessian, represents curvature for each parameter component p. In other words, the Levenberg-Marquardt method enables the computation by avoiding the singular matrix issue of Gauss-Newton, while also effectively reflecting the curvature even when μ is large so that the model can identify the global maxima effectively [13].

C. DDPG

RL (Reinforcement learning) is a value-based algorithm, and it learns by estimating the Q-value that the model will take in the given circumstance [14]. When estimations of this value become somewhat possible, actions (i.e., a policy) are chosen based on this value. Furthermore, Q-Learning has an e-greedy policy which estimates the value for all actions and then selects the action that corresponds to the largest number of those values. However, learning is not an easy task if there are many behaviors in the continuous action space [15].

DDPG (Deep Deterministic Policy Gradient) is a model-free off-policy actor-critic algorithm that combines DQN (Deep Q Network) with DPG (Deterministic Policy Gradient) [16], [17]. In the case of DQN, it reduces learning instability using the 'experience replay' and the 'frozen target network.' Typical Q learning obtains the data by the agent moving actually. Thus, naturally, there is a significant correlation between the data [18]. Therefore, the experience replay method is used to reduce the correlation between input data, significantly reducing the relationship among them. It also enables repetitive learning of past experiences [19].

The mathematical and numerical approaches based power system analysis require interpretation in continuous space, because changes in loads and generators' outputs are continuous instead of discrete. To this end, DDPG is a model that has the advantage of DQN, and can be extended to continuous space using the actor & critic framework. The critic framework is used to estimate the value using the Bellman equation. The actor framework is used to generate action according to the distribution of action space by the chain rule [20]. In addition, the DDPG normalizes the various unparticular units by normalizing the observation and uses batch normalization to put the samples into a single minibatch and normalize all the dimensions for better learning [19], [21]. The full algorithm, called DDPG, is presented in [20].

D. TD3

Q-learning algorithm is known to be affected by performance due to overestimation on the value function [22]. If the overestimation continues to occur during training, the policy update will be negatively affected. These features have led to the emergence of techniques called Double Q-Learning and Double DQN, which use two value networks to separate action selection updates and Q-value updates [23]. TD3 (Twin Delayed Deep Deterministic Policy Gradient) is an algorithm that applies several techniques to the DDPG for preventing overestimation on the value function.

The first technique is Clipped Double-Q Learning. TD3 learns two Q-functions instead of one (hence "twin"). Specifically, it means there are two actor networks, two critic networks, and two Bellman equations. The second technique is Delayed Policy Updates. TD3 updates the policy (and target networks) less frequently than the Q-function. That is, the model does not update policy unless the model's value function is updated sufficiently. These less frequent policy updates will have value estimate with lower variance and therefore should result in a better policy. This allows the value network to become more stable and reduce errors before it is used to update the policy network.

The third technique is Target Policy Smoothing. If the agent were to explore on-policy, it would probably not try a wide enough variety of action exploration to useful learning process. Unless there is an efficient noise in the environment, it is tough to ensure sufficient exploration to avoid the policy's determinacy. TD3 trains a deterministic policy in an off-policy way. By adding noise to the policy and learn with a non-deterministic policy, it can learn like off-policy [20]. TD3 adds noise (clipped random noise) to the target action and averaging over minibatches, as shown in Eq. (11): where r is a reward what we designate, and γ is a discount factor, which essentially determines how much the reward vector affect the reinforcement learning agents in the distant future relative to those in the immediate future, ranged from 0 to 1 [24]. Adding noise makes it harder for the policy to exploit Q-function errors by smoothing out Q along with changes in action.

$$\begin{aligned} y &= r+\gamma \mathrm{Q_w}\left(s', \mu_\theta\left(s'\right)+\varepsilon\right) \\ \varepsilon &\sim \mathrm{clip}\left(\mathrm{N}\left(0, \sigma\right), -\mathrm{c}, \mathrm{c}\right) \end{aligned} \tag{11}$$

The TD3 algorithm is an extended DDPG algorithm, and its computation is similar to the DDPG algorithm. Since TD3 has two critic models that induce loss function, critic loss is defined as MSE $\mathrm{Loss}(\mathrm{Q}_1(s, a'), \mathrm{Q_t}$ + MSE $\mathrm{LossQ}_2(s, a'), \mathrm{Q_t})$, where s is state and a' is actor network's target value including the random noise.

To determine the parameters of TD3, firstly backpropagating the critic loss, we update the parameters $\theta_\mathrm{i}$of the two critic models. And in every two iterations, we update the actor model's parameter φ by performing gradient ascent on the output θ_1 of the first critic model as in Eq. (12):

$$\begin{aligned} \theta_\mathrm{i} &\leftarrow \min_{\theta_\mathrm{i}} \mathrm{N}^{-1}\sum_{i=1}^{N}(y - \mathrm{Q}_{\theta_\mathrm{i}}(s, a))^2 \\ \nabla_\varphi \mathrm{J}(\varphi) &= \mathrm{N}^{-1}\sum \nabla_\mathrm{a} \mathrm{Q}_{\theta 1}(s, a)|_{\mathrm{a}=\pi_\varphi(\mathrm{s})} \nabla_\varphi \pi_\varphi(s) \end{aligned} \tag{12}$$

where N is $[s_t, a_t, r_t, s_{t+1}]$ from replay buffer. After that, we update φ_i, where φ and θ_1 are the weights of the parameter actor and the critic, respectively. Finally, we update the target networks as in Eq. (13), where θ is critic target, and φ is actor target, similar as the DDPG work by Polyak Averaging [25].

$$\begin{aligned} \theta'_\mathrm{i} &\leftarrow \tau\theta_\mathrm{i} + (1-\tau)\,\theta'_\mathrm{i} \\ \varphi'_\mathrm{i} &\leftarrow \tau\varphi_\mathrm{i} + (1-\tau)\,\varphi'_\mathrm{i} \end{aligned} \tag{13}$$

III. PROPOSED MODEL

A. DRL TD3 USING DNN LEVENBERG MARQUARDT

The TD3 agent receives the state input and decides a corresponding action vector according to the e greedy method by the reward vector. The TD3 algorithm uses two value networks to separate action selection update and Q-value update. Thus, we build two neural networks for the critic models and two for the critic targets. And we use a delayed update of the actor network, only updating it every 2-time steps instead of after each time steps, it can make more stable and efficient training.

To consider ramp-rate constraints in this article, we add a limiter as a comparator to restrict output difference between previous time step value P_G^{t-1} and current time step value $\ddot{P}_G^t$. It is possible to take into account since the TD3 model is used for environment with continuous action spaces. One episode consists of a number of steps (user defined), and for each step, the action and state according to action is stored in the episode storage. The information from the (t-1)-th step, which is already stored in the episode storage, can be called up (state, action, and so on), and it is possibly used to manage the current step. We can call up P_G of the (t-1)-th step, and thus we can set the limiter value at the t-th step of P_G in consideration of ramp rate. At this time, comparator compares P_G^{t-1} with P_G^t. If P_G^t meet the ramping constraints, the output should be the action that is precalculated in the proposed model as estimate. Otherwise, we change the current step's action to ($P_G^{t-1}\pm$ ramp-rate). It defines, when $\ddot{P}_G^t$ does not meet ramping constraints, P_G^t is updated as ($P_G^{t-1}\pm$ ramp-rate) instead of $\ddot{P}_G^t$. Moreover, considering inequality constraints on generation outputs, if P_G^t is greater than the $P_{Gi}^{t\ \max}$, P_G^t will be set to $P_{Gi}^{t\ \max}$. The proposed algorithm is shown in Figure 1:

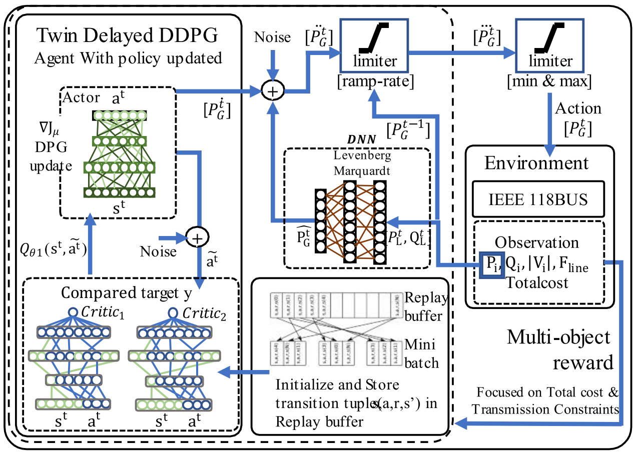

FIGURE 1. Workflow of the proposed approach.

To operate reinforcement learning, users must properly define action, state, reward, and environment. In the proposed model, the state includes amplitudes of the voltage and power data for all buses, which is formed as follows:

$$s_t = \left[P_L^t, Q_L^t, V_i^t, P_G^t, P_G^{t-1}, c, F\right] \tag{14}$$

where P_L^t means the active power load vector, Q_L means reactive power load vector, V_i^t is the bus voltage magnitude vector, F means power flow in each line, c is total cost according to load conditions of buses, P_G^t and P_G^{t-1} are generator's outputs in the current and previous steps for considering ramp-rate limits. Since TD3, which policy gradient methods used, is well known as a model to avoid the curse of dimensionality, although their units are not the same, TD3 algorithm does not need a procedure of state discretization like DDPG [6].

The load changes would potentially lead to changes on terminal voltages of the generator side. Thus, the constant terminal voltage of the generation side is preferred to ensure reliability of the system. To this end, action of the proposed approach is defined as active power outputs of generators. To decide the initial values of the action, we follow DNN's output P^t_G as TD3 agent's initial values when DNN with input $[P^t_L, Q^t_L]$ is trained based on Levenberg-Marquardt backpropagation. In other words, we set the DNN model's output as the initial value of the action with respect to certain load status so that the TD3 algorithm reaches the global optimum. After that, to make TD3 policies explore better, we add noise to the action vector (sum of actor network output and DNN output) when we train the model, uncorrelated mean-zero gaussian noise, and we reduce the scale of the noise throughout the training.

B. MULTI-OBJECT REWARD

ACOPF deals with several objectives, and the corresponding reward strategy is crucial. The major advantage of using reinforcement learning instead of DNN only in the proposed model is that generator's constraints such as P^{max}_G, P^{min}_G, and ramp-rate limits can be included in the learning process. Suppose P^t_G (i.e., action) of the next step exceeds the ramping constraint compared to the current step, the size of the action of the next step can be adjusted to satisfy constraints, and the output constraints of each generator can also be resolved by specifying a range in the model's action. Because it is possible to setup this environment in the programming work (i.e., user define), we do not need to set these constraints as reward.

As reward impacts TD3 algorithm's two Bellman equation (y_1, y_2), organizing a proper reward strategy is crucial to achieve better performance. However, if the policy changes slowly, the two networks in the TD3 algorithm become too similar to make independent decisions. To overcome this issue, TD3 algorithm uses the smaller of the two Q-values to form the targets in the Bellman error loss functions, as shown in Eq. (15). And γ (discount factor) applies equally to both critic networks.

$$\begin{aligned} \mathrm{y}_1 &= r+\gamma \min\nolimits_{\mathrm{i}=1,2} \mathrm{Q}_{\mathrm{w}2}(s', \mu_{\theta i}(s')) \\ \mathrm{y}_2 &= r+\gamma \min\nolimits_{\mathrm{i}=1,2} \mathrm{Q}_{\mathrm{w}1}(s', \mu_{\theta i}(s')) \end{aligned} \tag{15}$$

We offer a multi-object reward system, and form the reward vector as a sum of 4 components:

$$r = Reward1 - Reward2 - Reward3 - Reward4 \tag{16}$$

The main objectives in ACOPF problem are to minimize the total generation costs as 'the smaller, the better' problem. If we already know the cost function of generators, we easily calculate the total cost through the action vector (P^t_G). *Reward*1 takes into account the difference when the total cost becomes smaller according to the action. It makes a proposed model's policy that allows action to incur as little cost as possible under constraints. Also, the ACOPF shall consider the power system transmission congestion, since transmission lines have rated capacities and exceeding these capacities could lead to insecure operations. So, *Reward*2 expresses penalty as $max(Fe, 0)$, where Fe is the value of the difference between F and designated line capacities.

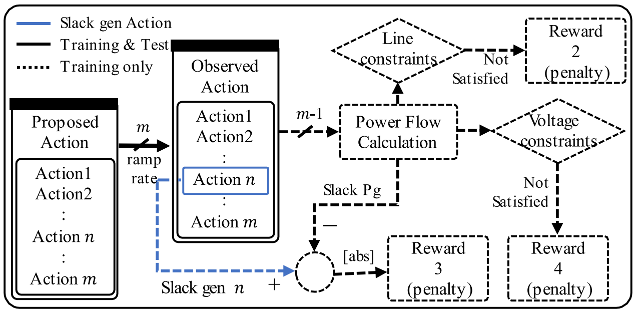

FIGURE 2. Take action by circumventing the penalty.

Slack generator's output is derived through the Newton-Raphson power flow calculation once outputs of the remaining generators are determined. So, we need to calculate power flow in every moment for power balance equation. However, real-time OPF shall be very fast in computation. If power flow calculation is needed per step, even testing under new load conditions, this is very inefficient in terms of computational time.

*Reward*3 is inversely proportional to absolute value of $(actual\ P_G\ of\ Slack\ bus - Action\ of\ Slack\ bus)$. We use power flow calculation when we train the model; but when testing the model, we neglect power flow calculation. When we examine the model in various load conditions, it is possible to ignore power flow calculation if the error of slack generator's output is smaller than the designated tolerance. If the model were well made, this error would be a small value because we set the *Reward*3 as a penalty to meet power balance equation. Also, *Reward*4 is a penalty proportional to the number of violations when nodal voltage magnitudes are not satisfied with voltage constraints in accordance with action vector.

Figure 2 contains the solution that the proposed model complements these mentioned matters of electrical constraints by taking proper reward as penalty. In Figure 2, we assume that there are m generators and nth generator is Slack generator. We have also adjusted reward vectors related to the constraints equally by normalization since they are measured in different units.

IV. PERFORMANCE ANALYSIS

A. OBJECTIVE AND DATASET

We use the IEEE 118-bus with 19 generators, 35 condensers, and 99 load buses for an experiment. Data on line impedance and system information, including cost function parameters, are referenced in IEEE 118-bus. Our goal is to ensure that the proposed approach meets constraints and produces optimal costs under various load conditions. The dynamic load profile shown in Figure 3 is used to evaluate that the ramp-rate limits are met.

Varying load changes can prove that the proposed model is robust against disturbance. This article's proposed model assumes that net load (both active power and reactive power) changes between 0.8 and 1.2, assumed the loads presented

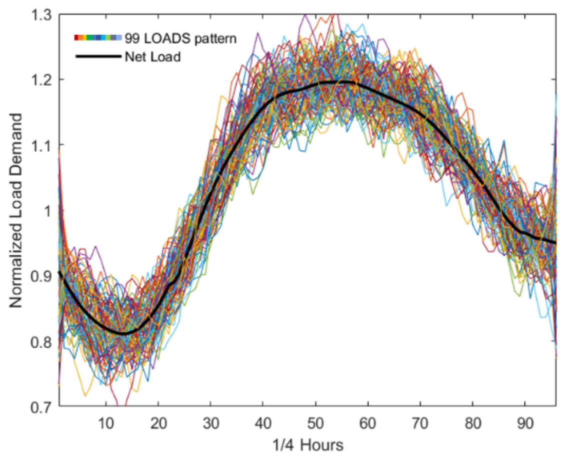

FIGURE 3. Load profile - 99 loads profile and net load (sum total).

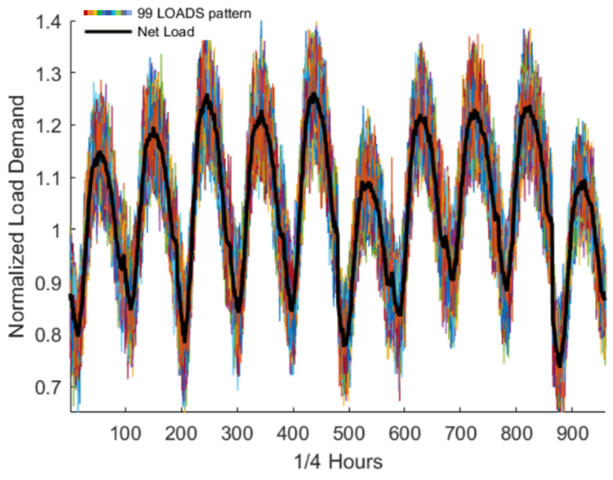

FIGURE 4. Data set organized.

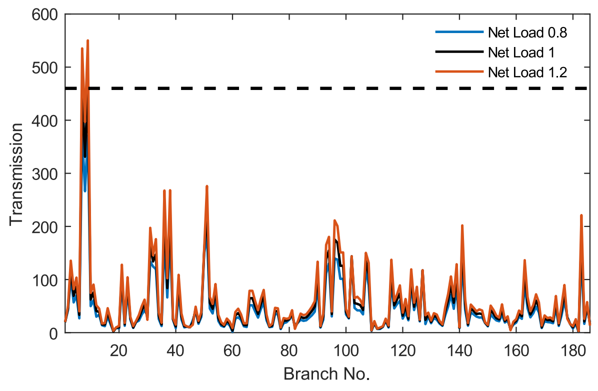

FIGURE 5. Branch Power flow not considered constraints.

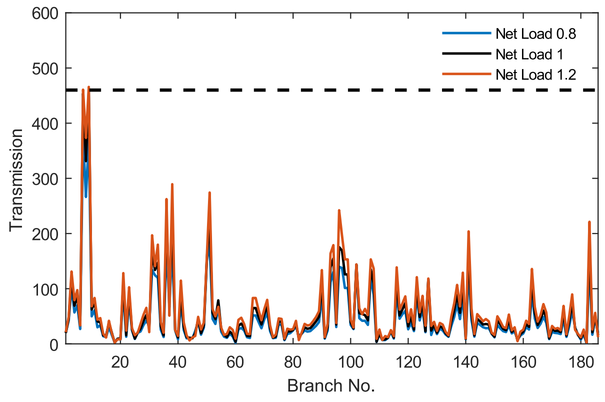

FIGURE 6. Branch power flow with transmission constraints.

in IEEE 118-bus Data Sheet as 1 p.u. In addition, to represent uncertainties of intermittent renewable energy sources in individual buses, we assume that each of the 99 loads has a random pattern that follows Gaussian noise.

We use the assumed Net Load curve to create a random load curve. As we will consider a 1/4 ramp-rate, we can see in Figure 4 that there are 960 samples for 10 days. 960 dispatch intervals of these data are set as the training set. Also, we made 10 test sets (960 $\times$ 10) dispatch intervals more, in the same way, to test the proposed approach with different load conditions. These results are compared with MAT-POWER MIPS to illustrate performance and robustness of the proposed model under various load conditions.

B. SYSTEM INFORMATION

At each step, we first set the power load according to our load design function. As described in Section IV.A, the performance of the proposed model is tested when the normalized Net Load fluctuates from 0.8 to 1.2, both active power and reactive power. The branch flow limits to 460 MVA, and generator ramping rates are set to 20 MVA.

Figure 5 shows the power flows of 186 branches while ignoring transmission constraints in the IEEE 118-bus system. In Figure 5, 460 MVA flows in one of the branches, when the normalized net load is 1. It means it might lead to transmission congestion if net load is greater than 1. Therefore, as we set the transmission constraints to 460 MVA in the test simulation, to check if the proposed model can properly handle both cases, transmission congestion happened and not.

C. TEST RESULTS

After the training is done, we use a new load pattern to demonstrate advantages of the proposed method by comparing it with two other methods: AC OPF (MIPS) and DC OPF (MIPS). AC OPF and DC OPF are realized using MAT-POWER [26]. The branch flow considering constraints are shown in Figure 6.

Figure 7 compares the outputs with and without considering ramp-rate constraints. Down ramp-rate constraint [-20 MVA] and each generator's ramp-rate of intervals from 670 to 674 are illustrated. Among the 960 time intervals, the 672nd step's ramp-rate is one of the examples to confirm that the proposed model can satisfy constraints. The left side of Figure 7 is the result of MAT-POWER ACOPF without considering any electrical constraints, and the right side of

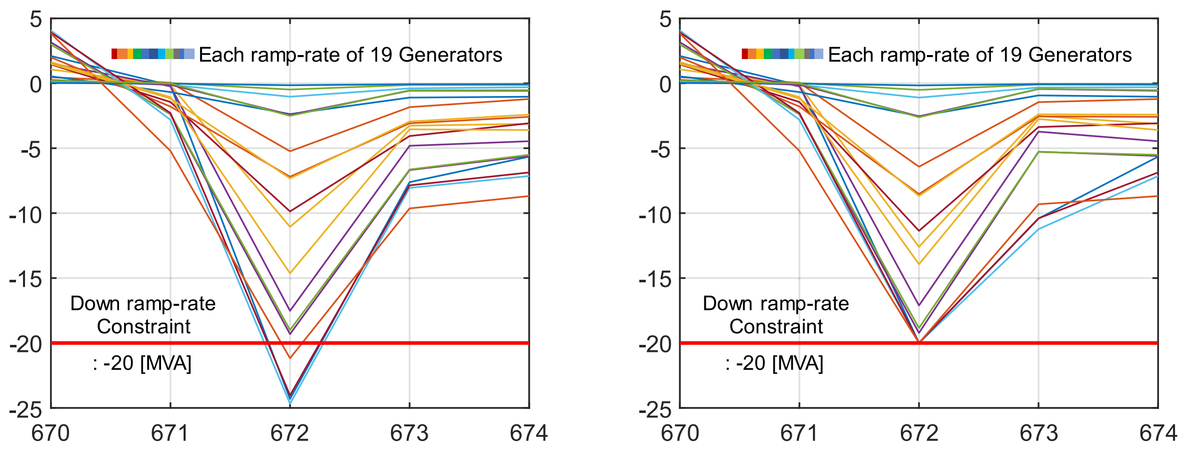

FIGURE 7. One example: ramp-rate of 19 Generators among the 960 intervals under Down ramp-rate constraint. [y-axis: ramp-rate.]

TABLE 1. Result comparison with MAT-POWER MIPS.

Methods comparison (TD3 with DNN)	Average generation cost (USD)	Average absolute errors of P_G (MW)	Inequality Constraints
AC OPF MIPS (MAT-POWER)	1.3843e5	0	All satisfied
DC OPF MIPS (MAT-POWER)	1.3269e5	11.2754	Branch flow and nodal voltage not satisfied
Proposed method	1.3843e5	0.0308	All satisfied

Figure 7 is the result of the proposed model considering ramp-rate constraints under the load profile (Section IV.A). As mentioned in Section III.A, the 672nd step's ramp-rate is contingent on the observed output of 671st step, P_G^{671}(the previous action). If the generators do not meet the ramp-rate constraints, P_G^{672} are determined from the modified ramp-rate. It shows that generators can meet ramp-rate constraints with the measure (limiter) in the proposed model.

Table 1 shows the detailed numeric comparison results, which compares the average value of the total generation cost of 960 real-time OPF intervals, the average absolute errors of the active power of generator between optimal solutions from different methods. Specifically, we calculate the exact slack P_G from power flow calculation using the remaining 18 P_G from the solution of the proposed model and check if the results are the same.

In our experiment, on average to compute the OPF problem with a given load profile, it takes 0.3017s [ACOPF-MIPS], 0.0452s [DCOPF-MIPS], and 0.0118s [Proposed Approach], respectively. And this computation has executed on a laptop with a 2.6-GHz Intel core i7-6700. This outcome explains that the proposed model is faster than DC OPF in terms of computing speed, and is capable of conducting economic dispatch planning under electrical constraints despite various arbitrary load profiles.

V. CONCLUSION

Using TD3 algorithm, this article carries out the OPF to minimize total cost under various constraints and illustrates robustness of the proposed model against disturbances(load fluctuations), where both active power and reactive power loads are assumed to vary between 0.8 and 1.2 p.u. We apply Gaussian noise to each net load for representing uncertainties of renewable energy sources. This training continues thoroughly under the off policy so that the result may be slightly different from MAT-POWER. But still, it matches almost 100% even if the load variation is severe compared to other methods in [6], [7].

Considering that the power system is more complicated these days, the computational performance of OPF also has to be fast enough for tracing load changes. In the future, we plan to develop the machine learning model that considers all various constraints and unit-commitment planning together.

REFERENCES

[1] A.-A.-A. Mohamed, Y. S. Mohamed, A. A. M. El-Gaafary, and A. M. Hemeida, "Optimal power flow using moth swarm algorithm," *Electric Power Syst. Res.*, vol. 142, pp. 190–206, Jan. 2017.

[2] K. Rahbar, J. Xu, and R. Zhang, "Real-time energy storage management for renewable integration in microgrid: An off-line optimization approach," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 124–134, Jan. 2015.

[3] S. Huang and V. Dinavahi, "Fast batched solution for real-time optimal power flow with penetration of renewable energy," *IEEE Access*, vol. 6, pp. 13898–13910, 2018.

[4] T. Jayabarathi, T. Raghunathan, B. R. Adarsh, and P. N. Suganthan, "Economic dispatch using hybrid grey wolf optimizer," *Energy*, vol. 111, pp. 630–641, Sep. 2016.

[5] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2963–2973, Nov. 2017.

[6] H. Nie, Y. Chen, Y. Song, and S. Huang, "A general real-time OPF algorithm using DDPG with multiple simulation platforms," in *Proc. IEEE Innov. Smart Grid Technol.-Asia (ISGT Asia)*, May 2019, pp. 3713–3718.
[7] Z. Yan and Y. Xu, "Real-time optimal power flow: A lagrangian based deep reinforcement learning approach," *IEEE Trans. Power Syst.*, vol. 35, no. 4, pp. 3270–3273, Jul. 2020.
[8] X. Pan, T. Zhao, and M. Chen, "DeepOPF: Deep neural network for DC optimal power flow," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Oct. 2019, pp. 1–6.
[9] W. F. Tinney and C. E. Hart, "Power flow solution by Newton's method," *IEEE Trans. Power App. Syst.*, vol. PAS-86, no. 11, pp. 1449–1460, Nov. 1967.
[10] X. Yu, M. O. Efe, and O. Kaynak, "A general backpropagation algorithm for feedforward neural networks learning," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 251–254, Aug. 2002.
[11] B. K. Singh, K. Verma, and A. S. Thoke, "Adaptive gradient descent backpropagation for classification of breast tumors in ultrasound imaging," *Procedia Comput. Sci.*, vol. 46, pp. 1601–1609, 2015.
[12] E. Mizutani and S. E. Dreyfus, "Second-order stagewise backpropagation for hessian-matrix analyses and investigation of negative curvature," *Neural Netw.*, vol. 21, nos. 2–3, pp. 193–203, Mar. 2008.
[13] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.
[14] L. C. Baird, III., and A. W. Moore, "Gradient descent for general reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 968–974.
[15] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5174–5184, Nov. 2018.
[16] J. Li, T. Chai, FL. Lewis, Z. Ding, and Y. Jiang, "Off-policy interleaved *Q*-learning: Optimal control for affine nonlinear discrete-time systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1308–1320, May 2019.
[17] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 1–9.
[18] M. Ramicic and A. Bonarini, "Correlation minimizing replay memory in temporal-difference reinforcement learning," *Neurocomputing*, vol. 393, pp. 91–100, Jun. 2020.
[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971
[21] X. Wu, S. Liu, T. Zhang, L. Yang, Y. Li, and T. Wang, "Motion control for biped robot via DDPG-based deep reinforcement learning," in *Proc. WRC Symp. Adv. Robot. Automat. (WRC SARA)*, Aug. 2018, pp. 40–45.
[22] HV. Hasselt, "Double Q-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2613–2621.
[23] J. Pan, X. Wang, Y. Cheng, and Q. Yu, "Multisource transfer double DQN based on actor learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2227–2238, Jun. 2018.
[24] W. B. Knox and P. Stone, "Reinforcement learning from human reward: Discounting in episodic tasks," in *Proc. IEEE RO-MAN: 21st IEEE Int. Symp. Robot Human Interact. Commun.*, Sep. 2012, pp. 878–885.
[25] S. Dankwa and W. Zheng, "Twin-delayed DDPG: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent," in *Proc. 3rd Int. Conf. Vis., Image Signal Process.*, Aug. 2019, pp. 1–5.
[26] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.

JONG HA WOO received the B.S. degree in electrical engineering from Konkuk University, Seoul, South Korea, in 2020. He is currently pursuing the master's degree in electrical engineering, Konkuk University, Seoul, under the supervision of Prof. Jae Hyung. Roh. His current research interests include the optimal power flow using a deep reinforcement learning, smart grid, and power system operation.

LEI WU (Senior Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in systems engineering from Xi'an Jiaotong University, Xi'an, China, in 2001 and 2004, respectively, and the Ph.D. degree in electrical engineering from the Illinois Institute of Technology (IIT), Chicago, IL, USA, in 2008. From 2008 to 2010, he was a Senior Research Associate with the Robert W. Galvin Center for Electricity Innovation, IIT. He was a summer Visiting Faculty with NYISO in 2012. He was a Professor with the Department of Electrical and Computer Engineering, Clarkson University, Potsdam, NY, USA, until 2018. He is currently a Professor with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA. His research interests include power systems operation and planning, energy economics, and community resilience microgrid.

JONG-BAE PARK (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Seoul National University, South Korea, in 1987, 1989, and 1998, respectively. From 1998 to 2001, he was with the Department of Electrical and Electronics, Anyang University, South Korea, as an Assistant Professor. From 2006 to 2008, he was a Resident Researcher with EPRI, USA. Since 2001, he has been with the Department of Electrical Engineering, Konkuk University, Seoul, South Korea, as a Professor. His major research topics include power system operation, planning, economics, and markets.

JAE HYUNG ROH (Member, IEEE) received the B.S. degree in nuclear engineering from Seoul National University, Seoul, South Korea, in 1993, the M.S. degree in electrical engineering from Hongik University, Seoul, in 2002, and the Ph.D. degree in electrical engineering from the Illinois Institute of Technology, Chicago, IL, USA, in 2008. From 1992 to 2001, he was with Korea Electric Power Corporation. From 2001 to 2010, he was with Korea Power Exchange. Since 2010, he has been with the Department of Electrical and Electronics Engineering, Konkuk University, Seoul, as a Professor. His research interests include electricity market, smart grid, and resource planning. He was a recipient of the IEEE PES Technical Committee Prize Paper Award in 2015.