Power System Flow Adjustment and Sample Generation Based on Deep Reinforcement Learning

Shuang Wu, Wei Hu, Zongxiang Lu, Yujia Gu, Bei Tian, and Hongqiang Li

***Abstract*—With the increasing complexity of power system structures and the increasing penetration of renewable energy, the number of possible power system operation modes increases dramatically. It is difficult to make manual power flow adjustments to establish an initial convergent power flow that is suitable for operation mode analysis. At present, problems of low efficiency and long time consumption are encountered in the formulation of operation modes, resulting in a very limited number of generated operation modes. In this paper, we propose an intelligent power flow adjustment and generation model based on a deep network and reinforcement learning. First, a discriminator is trained to judge the power flow convergence, and the output of this discriminator is used to construct a value function. Then, the reinforcement learning method is adopted to learn a strategy for power flow convergence adjustment. Finally, a large number of convergent power flow samples are generated using the learned adjustment strategy. Compared with the traditional flow adjustment method, the proposed method has significant advantages that the learning of the power flow adjustment strategy does not depend on the parameters of the power system model. Therefore, this strategy can be automatically learned without manual intervention, which allows a large number of different operation modes to be efficiently formulated. The verification results of a case study show that the proposed method can independently learn a power flow adjustment strategy and generate various convergent power flows.**

***Index Terms*—Deep reinforcement learning, power flow adjustment, system operation mode, sample generation.**

I. Introduction

THE operation modes of a power system are the overall technical schemes for power system production and operation formulated by the power dispatching department. As the scale of power grids, the proportion of renewable energy access and the diversity of power generation have increased, the modes of operation and other characteristics of power systems have also undergone significant changes [1]. Given the structural parameters and load conditions of a power grid, a reasonable operation mode must be determined in terms of the on-off modes of the generators, which must satisfy the requirement that the power flow calculation result is convergent. For a complex power system, it is very difficult to find a reasonable way to coordinate the outputs of the generators [2], and an excessive load or an unreasonable parameter configuration will usually lead to difficulty in reaching power flow convergence [3]. The traditional method of operation mode formulation for a power system is based on manual adjustment; specifically, each generator is manually adjusted in accordance with the operator's experience to obtain the initial power flow for an operation mode. However, this method of power flow adjustment is labor-intensive and inefficient, and cannot adapt to the complex and variable conditions encountered during actual operation.

The basic premise of power system operation mode formulation is to achieve convergent power flow conditions, and the purpose of power flow adjustment is to establish a convergent power flow state that basically satisfies the operation requirements of the power system. Reference [4] has found that as the load increases, the number of power system flow solutions is reduced in pairs until there is no solution. It is common to set initial values of active power and reactive power for the generators before calculating the power flow. If these initial values are set rationally, the power flow solution will converge. The set of initial values that can yield a convergent flow solution formulates the feasible region. When there is no power flow solution, it is necessary to adjust the power flow equation into the feasible region, i.e., adjust the initial active and reactive power flows of the generators. Therefore, the focus of power flow adjustment is to adjust a power flow scenario with no solution to a solvable one in as few steps as possible. Traditional power flow adjustment methods include sensitivity methods and nonlinear programming methods [5]-[10]. Reference [5] has proposed and defined an index for measuring the degree of insolvability of the power flow equation. The sensitivity to various means of adjustment can be calculated on the basis of this index, and the power flow can then be adjusted in accordance with this sensitivity. The sensitivity adjustment meth-

Manuscript received: April 17, 2020; accepted: September 22, 2020. Date of CrossCheck: September 22, 2020. Date of online publication: November 26, 2020.

This work was supported by the Science and Technology Project of the State Grid Corporation of China (No. 5400-201935258A-0-0-00), and the National Natural Science Foundation of China (No. 51777104).

S. Wu, W. Hu (corresponding author), and Z. Lu are with the Department of Electrical Engineering, Tsinghua University, Beijing, China (e-mail: wus16@mails.tsinghua.edu.cn; huwei@mail.tsinghua.edu.cn; luzongxiang98@tsinghua.edu.cn).

Y. Gu, B. Tian, and H. Li are with State Grid Ningxia Electric Power Co. Ltd., Yinchuan, China (e-mail: guyujia@nc.sgcc.com.cn; tianbei@nx.sgcc.com.cn; lihongqiang@nx.sgcc.com.cn).

DOI: 10.35833/MPCE.2020.000240

od proposed in [6] is applicable to voltage collapse caused by a large disturbance or an increase in load power of node. In [7], a sensitivity index similar to that in [5] is adopted, and a method based on the left eigenvector is adopted to restore the solution to the power flow equation. In [8], the weak link of the system is first calculated, and the sensitivity of the voltage phase angle to the injected power at a node is also calculated. Then, the control method is selected in accordance with this sensitivity. In [9], a power flow problem with no solution is regarded as a nonlinear programming problem, and a reduction of the active power levels of all buses is used as the objective function. In this nonlinear programming problem, the power flow equation serves as an equality constraint, and the control and state variable constraints are imposed as inequality constraints. Finally, the problem is solved using the interior point method. In [10], the load level is first reduced, and the characteristic index of the system is calculated to identify the weak link. Then, the sensitivity of each generator and each reactive power compensation equipment to the power flow on the weak link is calculated. The load level is restored by adopting adjustment measures on the basis of this sensitivity.

The above studies serve as a reference for the manual adjustment of power flows from the perspective of the power system mechanism, but the following problems remain:

1) For a large-scale power grid, the parameters of the system model are complex, and a high proportion of renewable energy access causes variable system characteristics, which makes it extremely difficult to explicitly express the sensitivity.

2) Using traditional methods, it is difficult to meet the high demand for power system operation modes, i.e., to efficiently generate a large number of convergent power flow samples.

References [11] and [12] propose approaches that can be used to automatically generate and adjust power flows, but limitations still remain. In [11], a DC flow model is used. In actual tests, convergence occurs, and this phenomenon is more obvious with the increasing scale of power grid. In [12], the flow adjustment procedure consists of splicing the flow in each region such that the flow of the whole system converges. The basic flow data for each region serve as the premise for this procedure. These approaches rely on system model data, and the adjustment procedure is not intelligent. Therefore, to overcome the problems posed by power flow adjustment for the operation modes of the current and future power systems, more intelligent and refined means of power flow adjustment are urgently needed to cope with the actual variable operation states of power systems.

As a comprehensive discipline and technology, artificial intelligence has enabled remarkable achievements in many research fields in recent years, and its performance is driven by basic theoretical research on big data and high-quality computer resources [13]. Deep learning, reinforcement learning and their combination have become popular topics in current research. Deep learning has achieved remarkable success in image analysis [14], speech recognition [15], natural language processing [16] and other fields, while deep reinforcement learning has reached or even surpassed the human performance in video games [17] and chess games [18]. With the development of smart grids and access to wind power and photovoltaic power at high proportions, the environment of power system operation has become increasingly uncertain and complex, and traditional analysis methods cannot meet the needs of future smart grid development [13]. As alternative approaches, deep learning and reinforcement learning offer significant advantages in dealing with complex problems such as classification, prediction, control and planning and have consequently received increasing attention in the field of power systems [19] for applications such as load forecasting [17]-[21], safety and stability control [22], [23], and economic dispatching [24]. A deep belief network (DBN) embedded with a parametric copula model is proposed in [20] for predicting hourly loads based on one-year data in urban Texas, USA. In [21], two long short-term memory (LSTM) models based on a neural network structure for forecasting hourly and minute-level loads are proposed, and the results show that an LSTM model based on the sequence-to-sequence approach is superior to a standard LSTM model for minute-level load prediction. In terms of security and stability control, [23] combines deep learning and reinforcement learning and adopts dual Q-learning and competitive Q-learning models to construct a cut-off strategy for power grid emergency control. Reference [23] also addresses a control problem. It proposes a practical semisupervised group prelearning method for the control performance standard (CPS) control strategy based on the Q-learning algorithm, which solves the problem of excessively fast convergence during the trial-and-error stage of the controller. In addition, [24] uses a combination of a tracking method and reinforcement learning to solve the economic dispatching problem. This algorithm has obvious advantages in terms of convergence flexibility and computation time. In recent studies, some scholars have tried to apply deep reinforcement learning method to power flow control and optimization. In [25], a simple Internet of Things system composed of one base station and multiple energy harvesting user equipment is studied. An actor-critic deep Q-network is proposed to deal with the access and continuous power control problem. This paper does not focus on power system level, although it uses state-of-art deep reinforcement learning. Aiming at the problem of coordination of distribution networks, [26] uses deep deterministic policy gradient method to manage power flow and control voltage of distribution networks. However, the scale of the case in this paper is small and it cannot prove the universality of the method. In [27], adaptive mapping strategy and Markov decision process (MDP) is formulated to solve the operation state calculation problem. Deep reinforcement method is trained to learn optimal adjustment strategy. However, this method still needs much manual intervention, and reactive power constraints are not considered. The proposed method can only deal with a certain tie-line, and cannot deal with various operation modes.

Power system operation mode is based on power flow adjustment. To realize the intelligent adjustment of power flow, this paper proposes a whole framework of intelligent including three parts: flow convergence discrimination, flow adjustment and flow generation, as shown in Appendix A Fig. A1. During the entire flow adjustment process, the environment model of power system is constructed. In flow adjustment part, the adjustment policy is learned fully automatically, and the same policy can also be used to generate convergent flows without much change. The whole framework is constructed combining the features of power system and artificial intelligence. The details are discussed in the next section.

In this paper, a three-part power flow adjustment framework to address power flow convergence, automatic power flow adjustment and sample generation is proposed. In all, the proposed framework uses deep learning and reinforcement learning method to adjust power flow more intelligently without much manual intervention, and can automatically generate various converged power flow samples. The remainder of this paper is organized as follows. Section II introduces the basic theories and models of deep learning and reinforcement learning. Section III describes the proposed core framework and analyzes the three parts of the framework in detail. In Section IV, the China Electric Power Research Institute (CEPRI) 36-bus and IEEE 118-bus test systems are used to demonstrate the proposed method. Finally, Section V presents the conclusions of the paper.

II. Deep Learning and Reinforcement Learning

A. Deep Learning and Neural Network

Deep learning is a specific learning method applied in artificial intelligence. As a perceptron model with multiple hidden layers, a deep neural network (DNN) is a typical model used for deep learning. Compared with a single-layer perceptron model, a fully connected neural network with multiple hidden layers has a more complex expression capability. Figure 1 shows the typical general structure of a DNN.

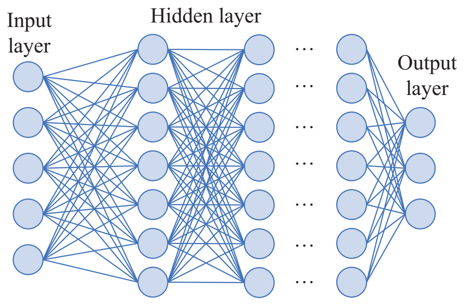

Fig. 1. Structure of a DNN: input layer, hidden layer, and output layer.

The information in the feedforward neural network is transmitted from the uppermost layer to the lower layers, and the output of each upper layer is the input to the next lower layer. The different layers of a DNN can be divided into an input layer, several hidden layers, and an output layer. The numbers of each layer and their neurons are specially designed. Suppose that $\boldsymbol{x}^{l-1}$ is the output of layer $l-1$; $\boldsymbol{W}^{l}$ is the weight matrix between layers l and $l-1$; $\boldsymbol{b}^{l}$ is the bias vector; and σ is the activation function. The output of layer l is:

$$\boldsymbol{x}^{l-1}=\sigma\left(\boldsymbol{W}^{l}\boldsymbol{x}^{l-1}+\boldsymbol{b}^{l}\right) \tag{1}$$

To enable neurons to process nonlinear data, continuous nonlinear functions such as the ReLU function, are typically used as activation functions. The activation function of the output layer is selected in accordance with the problem type (e.g., the sigmoid activation function is used for binary classification problems, and the softmax activation function for multiclass classification problems). Neural networks are generally trained using the backpropagation algorithm. For some deep-layer networks that are difficult to train, unsupervised pre-training combined with supervised fine-tuning can be used for further training [28]. In addition, the selection of the loss function is also related to the activation function.

B. Fundamental Theory and Model of Reinforcement Learning

Reinforcement learning is a kind of machine learning method that is different from both supervised learning and unsupervised learning. Unlike unsupervised learning, it considers reward values in addition to the data characteristics rather than the data characteristics alone. However, the reward values considered in reinforcement learning are also different from the labels used in supervised learning in that they are not specified before training but rather are generated only during the training process itself, after a certain delay. Table I summarizes the differences between reinforcement learning and supervised learning.

TABLE I
Comparison Between Reinforcement Learning and Supervised Learning

Supervised learning	Reinforcement learning
Labels required	Labels not required
Static training process	Dynamic training process
No feedback	Delayed feedback
Used for decision-making	Used for perception

The basic elements of a reinforcement learning model include the actions of the agent, the states of the environment and the rewards from the environment. The agent determines the current state of the environment and chooses an appropriate action in accordance with a certain strategy. The environment will then change its state and provide the current reward that results from the action. Then, the agent repeats the above process, thus interacting with the environment. Figure 2 shows a basic reinforcement learning model.

A reinforcement learning problem can be modeled as an MDP, which consists of four elements:

$$M=\{S,A,P,R\} \tag{2}$$

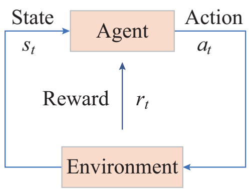

Fig. 2. Reinforcement learning model.

where S denotes the set of all states of the environment; A is the set of actions that the agent can execute, with a_t denoting the action that the agent executes at time t; P is the probability distribution function of the state transitions, with $P(s_t+1|s_t,a_t)$ denoting the probability of the environment transition from a state s_t to the next state s_t+1 after the agent executes action a_t; and R is the instantaneous reward, with $R(s_t,a_t)$ denoting the instantaneous reward provided to the agent after it executes action a_t in state s_t. Reinforcement learning constructs a mapping from environmental states to actions, which allows the agent to obtain its maximum accumulative reward through its interaction with the environment. R can be formulated as the accumulative reward from the start time t to the end time T, as shown in (3), where γ is a discount factor. Figure 3 illustrates the MDP concept.

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t \tag{3}$$

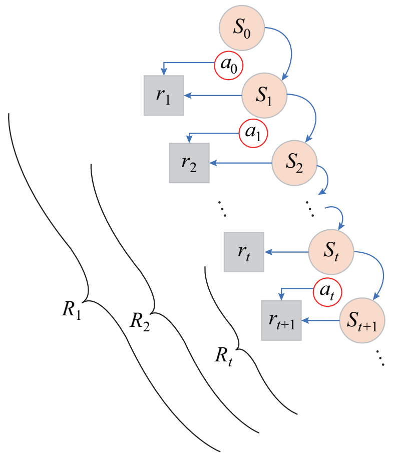

Fig. 3. MDP.

The ultimate goal of reinforcement learning is to learn the optimal strategy for executing actions, i. e., the probability with which action a should be executed in state s, as shown in (4). To solve for this optimal strategy, a state-action value function is defined, as shown in (5). G_t is the sum of all rewards (considering the discount factor) from a certain state to the termination state, as shown in (6). According to the expression for the state-action value function, a recursive formulation of this function, i.e., the Bellman equation, can be deduced as shown in (7).

$$\pi\left(a|s\right)=P\left(a_t=a|s_t=s\right) \tag{4}$$

$$Q^\pi\left(s,a\right)=E^\pi\left[G_t\middle|s_t=s,a_t=a\right] \tag{5}$$

$$G_t=\sum_{i=1}^{n}\gamma^{i-1}R_{t+i} \tag{6}$$

$$Q^\pi\left(s,a\right)=E^\pi\left(R_{t+1}+\gamma Q^\pi\left(s_{t+1},a_{t+1}\right)\middle|s_t=s,a_t=a\right) \tag{7}$$

where E^π is the mathematical expectaion of the reward value under policy π; and Q^π is the state-action value function of this reinforcement learning model.

Solving the reinforcement learning problem is equivalent to finding the optimal strategy π^*, and the optimal state-action value function corresponds to the most valuable one of all actions generated by all strategies. The optimal state-action value function and the optimal strategy based on the state-action value function are shown in (8) and (9), respectively.

$$Q^{\pi^*}\left(s,a\right)=\max_\pi Q^\pi\left(s,a\right) \tag{8}$$

$$\pi^*\left(a|s\right)=\begin{cases}1 & a=\operatorname{argmax} Q^*\left(s,a\right)\\ 0 & \text{else}\end{cases} \tag{9}$$

C. Temporal Difference Methods and State-action-reward-state-action (SARSA) Algorithm

For practical reinforcement learning problems, it is unrealistic to solve the state-action value function using the recursive iterative Bellman equation or a dynamic programming method because the state space of the actual environment is often very large, resulting in an enormous amount of calculation. In addition, in most cases, the state transition model of the environment, i.e., the state transition probability P, is unknown. Therefore, a model-independent method is usually required in practical cases. Monte Carlo and temporal difference are two typical kinds of model-independent methods. Unlike temporal difference method, Monte Carlo method requires the final, complete sequence of states, without which the problem cannot be solved. In addition, the variance of the actual cumulative return G is relatively large. Although the value obtained through temporal difference method is biased, it has a small variance and can be obtained ahead of the final result. In particular, it can be continuously learned in the environment. Therefore, the quantitative status estimates can be updated more quickly and flexibly. The iterative expression for the value function in a temporal difference method is shown in (10), where $\alpha\in(0,1)$.

$$Q\left(s,a\right)=Q\left(s,a\right)+\alpha\left(G-Q\left(s,a\right)\right) \tag{10}$$

The SARSA algorithm is one example of a temporal difference method. It includes five elements: S, A, R, γ, and the exploration rate ε. First, an action a is selected for the current state s based on the ε-greedy method, and the environment transitions to the next state s' and provides feedback in the form of the instantaneous reward R. Then, again based on the ε-greedy method, another action a' is selected for the state s' to update the value function. The formulas for the value function update and the ε-greedy method of the SARSA algorithm are shown in (11) and (12), respectively.

$$Q\left(s,a\right)=Q\left(s,a\right)+\alpha\left(R+\gamma Q\left(s',a'\right)-Q\left(s,a\right)\right) \tag{11}$$

$$\pi\left(a|s\right)=\begin{cases}\varepsilon/m+1-\varepsilon & a^*=\operatorname{argmax} Q^*\left(s,a\right)\\ \varepsilon/m & \text{else}\end{cases} \tag{12}$$

where $m=|A|$ is the number of actions in the action set; and a^* is the action that maximizes Q.

III. Intelligent Adjustment of Power System Flow

A. Overall Framework

The overall framework for intelligent power system flow adjustment designed in this paper includes three components: power flow convergence judgment, power flow adjustment, and power flow generation. The first component provides the necessary information for power flow adjustment. The second component provides an appropriate strategy for adjusting a nonconvergent power flow scenario to achieve convergence and guides the process of sample generation. This component is the focus of this paper. The third component is power flow sample generation, which corresponds to the final purpose of the framework. Figure 4 shows a schematic diagram of the overall framework.

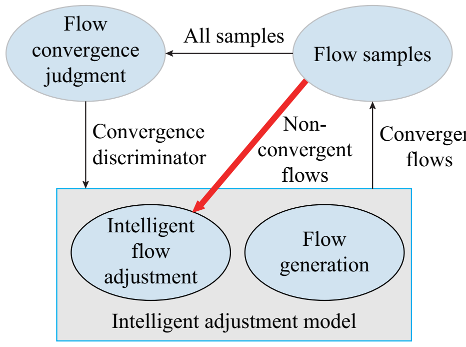
Flow convergence judgment
All samples
Flow samples
Convergence discriminator
Non-convergent flows
Convergent flows
Intelligent flow adjustment
Flow generation
Intelligent adjustment model

Fig. 4. Overall framework for intelligent flow adjustment.

The power flow convergence discriminator is trained with the samples in a flow database to judge convergence. The power flow adjustment component and the power flow generation component are closely related. The power flow discriminator provides information for the adjustment component, and the adjustment strategy trained by this component guides the power flow generation process to supplement the flow database with additional convergent power flow samples. The remainder of this section will introduce each component of the framework in detail.

B. Power Flow Convergence Judgement Based on DNN

The essential goal of power flow convergence judgment is to obtain a convergent solution to (13) with given initial power flow settings.

$$\frac{P_i - \mathrm{j}Q_i}{\dot{U}_i} = \sum_{j=1}^{n} Y_{ij}\dot{U}_j \quad i = 1, 2, ..., n \tag{13}$$

where n is the number of buses in the system; P_i, Q_i, and U_i are the active power, reactive power, and voltage of bus i, respectively; and Y_{ij} is the admittance of buses i and j.

Currently, mature commercial software can quickly perform power flow calculation and indicate whether power flow is converged. For example, MATPOWER can calculate the power flows for the IEEE standard test systems, and the power system analysis software package (PSASP) can calculate the power flow of a real system with more than 40000 buses. However, when such a commercial software package is fed with a set of initial power flow values, it can determine only whether the power flow scenario is convergent. For a nonconvergent flow scenario, this software cannot provide much information about how to adjust the initial settings to achieve convergence. Although in a small system, including slack variables in the formula can help to roughly identify the causes of nonconvergence, it is not realistic to build such a formula to describe a whole large power system, and therefore, the necessary adjustments must usually be made on the basis of the operator's knowledge and experience, not only with the assistance of commercial software.

If sufficient information can be obtained before flow convergence judgement, it will be more advantageous to set the initial value so as to make the power flow calculation convergent. Therefore, to judge flow convergence and provide additional information, this paper proposes the use of a DNN as a flow discriminator. This is a suitable approach because flow judgment is actually a binary classification problem. The output of a DNN is usually used only for category determination. However, since this output is probabilistic in nature, it can provide additional information about the different categories. For a binary classification problem, the number of dimensions of the DNN output layer is set to 1. As one of the most commonly used activation functions in binary classifiers, the sigmoid function is adopted here. The output result can be expressed as shown in (14).

$$f_y(x) = \frac{1}{1+e^{-x}} \tag{14}$$

The output is not a binary value but rather a continuous value between 0 and 1. Therefore, it can be regarded as a probability. This probabilistic output indicates how close the input is to the target category. As a flow convergence discriminator, the DNN takes the initial settings $\boldsymbol{X}$ (i.e., the active power P, reactive power Q, and voltage V of each generator) as input. The output of the DNN is the convergence probability of the input flow sample, as shown in (15), where $prob_i$ is the output and f_D is the DNN discriminator itself. The probability can be used in the reinforcement learning process.

$$prob_i = f_D(\boldsymbol{X}) \tag{15}$$

Before training the discriminator, the flow samples should be prepared in advance. In this paper, different basic flows are obtained by randomly setting the active and reactive power of generators and loads. Specifically, after selecting a base power flow, the generator output is set to be between 80%-120% of the current level, while load level is set to be 60%-120% of the current level. The convergence of flow is determined by flow calculation, and the flow sample is labeled according to the calculation result.

In the training process of discriminator, the adaptive moment estimation (Adam) optimization is used. The basic steps are as follows:

Step 1: initialize parameters θ, m_0, v_0.

Step 2: set $t = t+1$.

Step 3: update parameters:

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \tag{16}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{17}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{18}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{19}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{20}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{21}$$

where t is the timestep; m_t is the element of the first-moment vector; v_t is the element of the second moment vector; $\hat{m}_t$ and $\hat{v}_t$ are intermediate variables of the calculation; $f(\theta)$ is cross entropy loss function with θ; and α, β_1, and β_2 are preset parameters.

Step 4: if θ converged, end; otherwise, return to *Step 3*.

C. Power Flow Adjustment Model Based on Reinforcement Learning

Reinforcement learning is usually applied to solve optimization problems for time-related multi-stage decision-making processes. Power flow adjustment is not this kind of problem because if two consecutive actions are performed, the specific sequence of these two actions has no effect on the result of the power flow calculation. However, if a time factor is artificially introduced, power flow adjustment can also be regarded as a multi-stage decision-making process, and the final adjustment strategy can be obtained by combining the step-by-step strategies with the whole process.

To build a power flow adjustment model based on SARSA, a reinforcement learning model of the power system is required. Table II shows the correspondence between the elements of the power flow adjustment model and the elements of the reinforcement learning framework.

TABLE II
CORRESPONDENCE BETWEEN FLOW ADJUSTMENT MODEL AND REINFORCEMENT LEARNING

Reinforcement learning	Flow adjustment
Environment	Power system
State set	P and Q values of generators
Action set	Increases or decreases in controllable variables
Reward	Defined by the discriminator and other rules

The environment in the reinforcement learning framework corresponds to the power system itself, and the set of environmental states corresponds to the active and reactive power settings of the generators in the power system. Since the loads are constant for the power flow adjustment problem, the loads can be considered invisible to the environment. The state set is shown in (22), where k is the number of PQ node generators and l is the number of PV node generators.

$$\boldsymbol{S} = \left\{P_1, Q_1, P_2, Q_2, ..., P_k, Q_k, P_{k+1}, P_{k+2}, ..., P_{k+l}\right\} \tag{22}$$

The action set corresponds to the possible increases or decreases in the values of the controllable variables, such as increases in the active and reactive power of the PQ node generators or decreases in the active power of the PV node generators, as shown in (23), where m denotes the number of variables; X denotes a variable; and the subscript of X represents the concrete action applied to X. In a given action vector, only one variable is set to 1, and the rest are 0. The adjustment procedure consists of many steps, and in each step, if $X_{1,up}=1$, a specified value will be added to the variable X_1. The final action is equal to the sum of all actions in all steps.

$$A = \left\{X_{1,up}, X_{1,down}, X_{2,up}, X_{2,down}, ..., X_{m,up}, X_{m,down}\right\} \tag{23}$$

In the process of power flow adjustment, the instantaneous reward value provided as the environmental feedback after each action depends on the result of the power flow convergence judgment and the system state. The feedback reward is related to the output of the discriminator. The reward is greater if the discriminator gives a larger output. Furthermore, during power flow calculation, considering the influence of the reactive power balance, it is difficult for the power flow solution to converge when the difference between the reactive power of the generators and the reactive power of the loads is too large. Therefore, the degree of reactive power balance in the initial configuration should be considered when determining the reward value. The definition of the reactive power balance is shown in (24). The numerator is the total reactive power of the generators, and the denominator is the total reactive power of the loads.

$$B_q = \frac{\sum Q_{gen}}{\sum Q_{load}} \tag{24}$$

The instantaneous reward is shown in (25).

$$\begin{cases} R = R_1 + R_2 + R_3 \\ R_1 = \begin{cases} r_{11} & prob < p_1 \\ r_{12} & p_1 \le prob < p_2 \\ r_{13} & prob \ge p_2 \end{cases} \\ R_2 = \begin{cases} r_{21} & B_q < b_1 \text{ or } B_q > b_2 \\ r_{22} & b_3 \le B_q < b_4 \\ r_{23} & b_4 \le B_q < b_2 \end{cases} \\ R_3 = \begin{cases} r_{31} & \text{beyond limit} \\ r_{32} & \text{else} \end{cases} \end{cases} \tag{25}$$

where R_1 is the reward corresponding to the convergence result; R_2 is the reward corresponding to the reactive power balance. If the reactive power is relatively well balanced, the reward will be large; R_3 is the reward corresponding to the generator output limit. If the output of a generator exceeds its limit, a negative reward is returned to punish the previous action; and p_i and b_i are parameters depending on the specific characteristics of the problem.

D. Power Flow Adjustment Strategy Based on SARSA

The input to the SARSA algorithm includes the number of iterations T, the state set S, the action set A, the discount factor γ, the step size α, and the exploration rate ε. The output is the state-action value Q corresponding to all the states and actions. The learning process is as follows:

Step 1: randomly initialize all state-action values $Q(s,a)$.

Step 2: set $i=1$. If $i<T$:

1) Initialize s as the first state and a as the action selected via the ε-greedy method in state s;

2) Execute action a. The state s changes to s', and a re-

ward R is returned;

3) Select action a' in state s' in accordance with the ε-greedy method;

4) Update the state-action value $Q(s,a)$ according to (11);

5) Set s' as the current state s and a' as the current action a;

6) If s is the final state, end the current iteration and let $i=i+1$. Otherwise, return to 2).

The learning process for the power flow adjustment strategy is shown in Fig. 5.

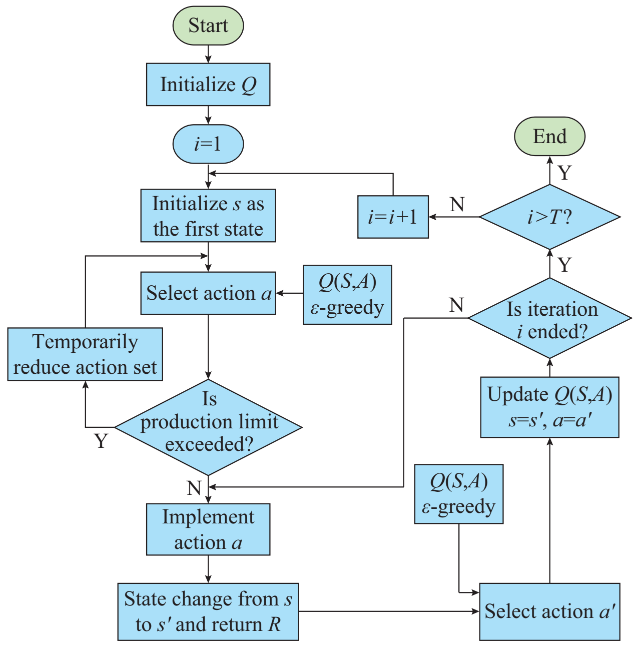
Start
Initialize Q
i=1
Initialize s as the first state
Select action a
$Q(S,A)$ ε-greedy
Temporarily reduce action set
Is production limit exceeded?
Y
N
Implement action a
State change from s to s' and return R
$Q(S,A)$ ε-greedy
Select action a'
Update $Q(S,A)$ $s=s'$, $a=a'$
Is iteration i ended?
N
Y
$i>T$?
N
Y
$i=i+1$
End

Fig. 5. Learning process for flow adjustment strategy.

E. Power Flow Generation Model Considering Uncertainty

The purpose of intelligent power flow adjustment is to generate a large number of convergent power flow samples by learning an appropriate adjustment strategy. In the field of video games, the environment of reinforcement learning is often uncertain or invisible. By contrast, for power flow adjustment, every definite action will lead to a definite change in the environmental state. Therefore, the reinforcement learning process may finally converge to a definite strategy, and thus, to a definite power flow state. To use the learned strategy to generate power flow samples, the power flow adjustment model is fine-tuned, and two types of uncertainty factors are added as follows: ① uncertainty of the environmental states; ② uncertainty of the actions.

Since the active and reactive power levels of the generators in the system are continuous variables, the actual number of environmental states is infinite. However, an infinite number of states cannot be considered in the SARSA algorithm. Based on the actual operation conditions of the power system, the output of each generator can be divided into three states, as shown in (26), where s_i is the i^{th} state variable dimension. The uncertainty of the actions is reflected in X_{i_up} and X_{i_down}. We set each value in a range in the form (a, b), e.g., (0.2,0.4).

$$s_i=\begin{cases} s_{i1} & 0\le s_i<0.5s_{\max} \\ s_{i2} & 0.5s_{\max}\le s_i<0.8s_{\max} \\ s_{i3} & 0.8s_{\max}\le s_i\le s_{\max} \end{cases} \tag{26}$$

By introducing uncertainty of the states and actions, the following effects are achieved: ① the number of environmental states is reduced, allowing reinforcement learning to be applied to the realistic situation; ② different instances of the same nominal environmental state may correspond to different actual operation states of the system; ③ the same strategy can produce different effects on the system and generate different power flow samples.

IV. Case Study

A. Case of CEPRI 36-bus System

1) System introduction and sample database

In this paper, the proposed method is verified based on the CEPRI 36-bustest system. A diagram of the CEPRI 36-bus test system is shown in Fig. 6.

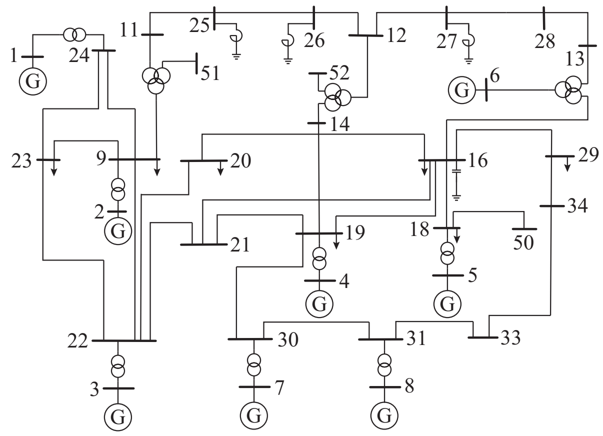

Fig. 6. Diagram of CEPRI 36-bus test system.

The system includes 8 generators, 32 lines and 10 loads. The generator outputs and loads have been randomly sampled, and the results are taken as the initial power flow settings while keeping the total active outputs of the generators and the loads approximately equal. A total of 81037 samples are generated, including 13185 convergent samples and 67875 nonconvergent samples. Hence, the proportion of convergent power flow samples is 16.3%. These samples are used to train the power flow convergence discriminator. The model training time is 17.7 min. The computer configuration is as follows: an i7-6700 CPU and 8 GB of RAM.

2) Power flow convergence discriminator

A total of 10000 convergent samples and 40000 nonconvergent samples are selected as the training set, and 3000 convergent samples and 12000 nonconvergent samples are selected as the test set. Tensorflow 1.3 is used to build the network. The DNN consists of one input layer, four hidden layers and one output layer. The numbers of neurons in each layer are 36, 16, 16, 8, 8, and 1, respectively. The input dimensions are determined by the numbers of generators and loads, and the output is the convergence probability of the input flow sample. To prevent overfitting during the training process, a dropout layer is included after each hidden layer. The Adam algorithm is used to optimize the loss function. The parameters are set to default values of Tensorflow. The learning rate is set to 0.001, and β_1 and β_2 are set to 0.9 and 0.99, respectively. To visualize the output result of each layer of the DNN, the *t*-SNE method is used to map the high-dimensional input space and the high-dimensional representation space of each hidden layer to a two-dimensional plane, as shown in Fig. 7. Figure 8 shows the changes in accuracy and loss value that have occurred during the training of the DNN.

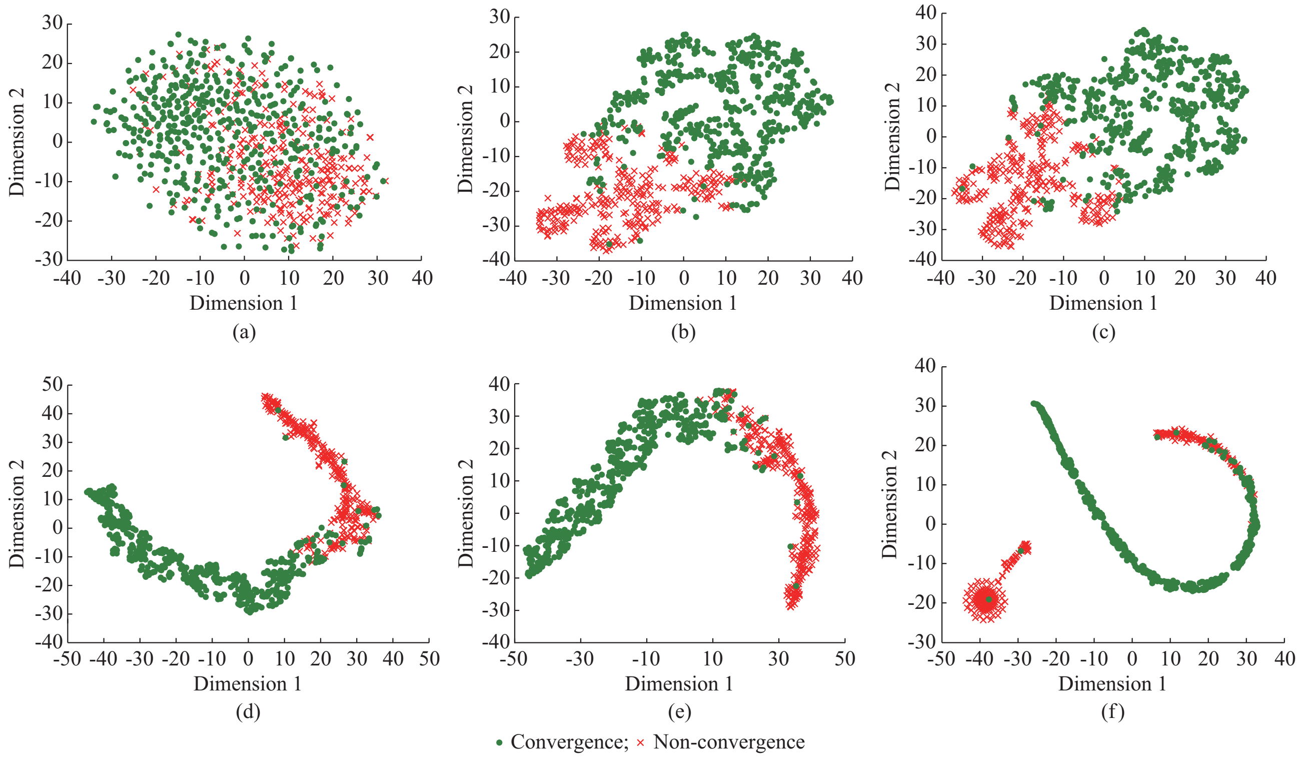

Fig. 7. Projections of internal space of DNN onto a two-dimensional plane. (a) Layer 0. (b) Layer 1. (c) Layer 3. (d) Layer 4. (e) Layer 5. (f) Layer 6.

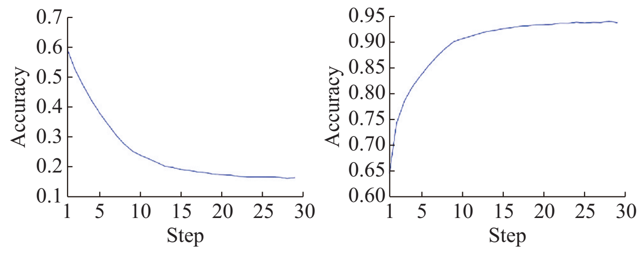

Fig. 8. Changes in accuracy and loss value during training.

It can be observed from Fig. 7 that the convergent and nonconvergent samples are clustered together in the original input space and are not naturally separated. However, in hidden layers of increasing depth, the DNN continues to learn increasingly high-dimensional features extracted from the original input, and these features are helpful for distinguishing the two types of samples. Finally, in the output layer, the two types of samples are almost completely separated. Thus, it is demonstrated that a simple classifier can achieve great results in the representation space. Figure 8 shows that with an increasing number of training iterations, the classification accuracy gradually increases, and the loss value gradually decreases. The accuracy reaches 93.6% at the 25th iteration, after which it no longer increases significantly. Thus, it is proven that the trained power flow discriminator can effectively distinguish between the power flow convergence and nonconvergence based on the initial settings.

3) Intelligence power flow adjustment

For this case study, two samples are randomly selected from the nonconvergent samples in the sample database for a power flow adjustment test. The specific data of these two power flow samples are shown in Appendix A Table AI. For the initial power flow settings, all voltage amplitudes are set to 1, and the reactive power levels of all PV node generators are not considered. Therefore, these parameters are omitted in the table. The parameters of SARSA process are as follows: initial step size α is set to 0.5; initial exploration rate ε is set to 0.5; discount factor γ is set to 0.2. The exploration rate gradually decreases in the training process.

For each nonconvergent power flow sample, the number of training iterations is set to 2500, and the number of actions required in each iteration of convergence adjustment is recorded. Figures 9 and 10 show the first 500 iterations of the training process, and Figs. 11 and 12 show the last 200 iterations of the training process. It can be seen from the above figures that during the initial stage of training for the power flow adjustment model, the numbers of actions required to achieve convergence are very large and vary greatly. As seen in Fig. 9, the number of actions at the 200th iteration is more than 1500; at the 201st iteration, it becomes approximately 800; and at the next iteration, it exceeds 1000 again, which indicates that the model still requires further training.

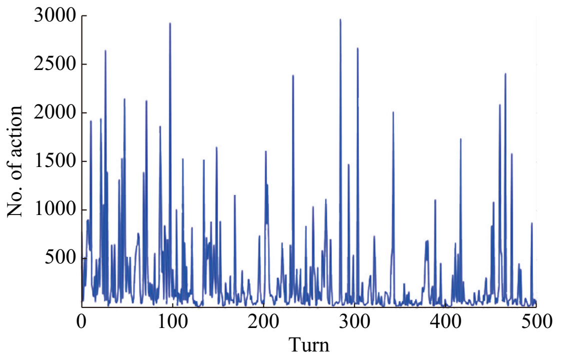

Fig. 9. First 500 iteration of sample 1.

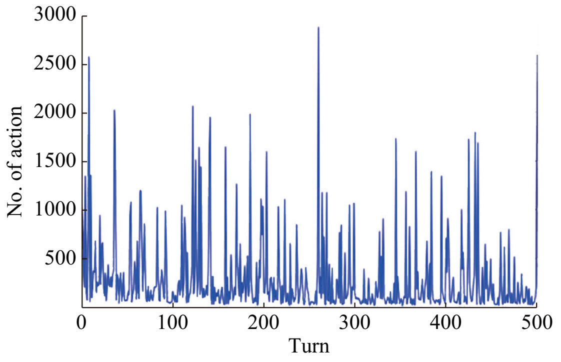

Fig. 10. First 500 iteration of sample 2.

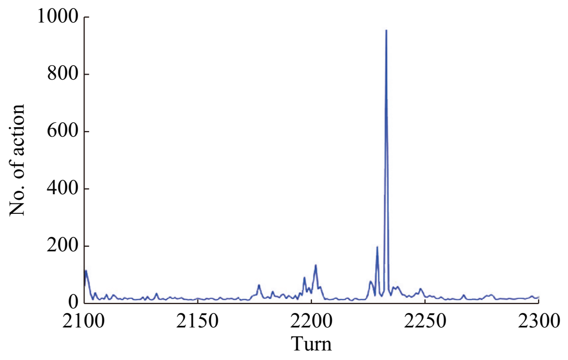

Fig. 11. Last 500 iteration of sample 1.

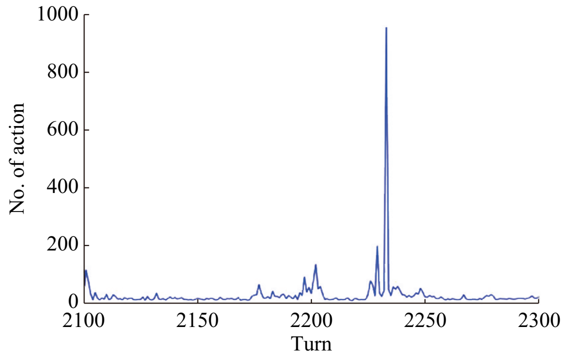

Fig. 12. Last 500 iteration of sample 2.

During the final stage of the training process, however, the number of actions remains relatively stable at approximately 20, which indicates that the state-action value function has converged and an appropriate adjustment strategy has been learned.

Nevertheless, it can be seen from Fig. 11 and Fig. 12 that at some iterations, the numbers of actions required are still much greater than 20. This can be attributed to the uncertainty introduced into the model and the characteristics of the ε-greedy algorithm itself. Although the uncertainty has some detrimental influence on the learning process, it is beneficial for the intelligent generation of convergent power flow samples. Table III presents the action sequences selected in the last three iterations of training for the adjustment model. There are two reasons for the differences in these action sequences: one is the ε-greedy method used in action selection, and the other is that the state-action function values of different actions may be the same.

TABLE III
Action Sequences Selected in Last Three Iterations

Iteration	Action No.
2293	7, 17, 14, 14, 14, 14, 7, 7, 11, 11, 6, 0, 16, 0, 0, 16, 16, 0, 16, 1, 1, 14, 15
2294	7, 14, 14, 14, 2, 16, 16, 16, 16, 7, 1, 16, 0, 16, 14, 15
2295	11, 7, 14, 14, 14, 17, 14, 14, 11, 0, 16, 16, 16, 16, 16, 17, 16, 16, 4, 2, 2, 15

Figures 13 and 14 show the change of average reward of each turn during the training process of the proposed model.

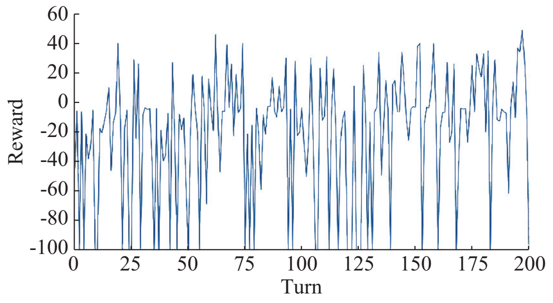

Fig. 13. Average reward for first 200 iterations.

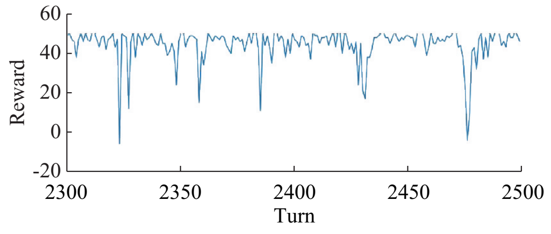

Fig. 14. Average reward for last 200 iterations.

According to (19) and the case, the parameters are set as follows: $p_1 = 0.8$, $p_2 = 0.9$, $r_{11} = -30$, $r_{12} = 30prob$, $r_{13} = 50$; $b_1 = 0.6$, $b_2 = 1.4$, $b_3 = 0.8$, $b_4 = 1.2$, $r_{21} = -80$, $r_{22} = -20 \times (1 - B_q)$, $r_{23} = 0$; $r_{31} = -100$, $r_{32} = 0$. The setting of negative parameters is to give penalty to reactive power imbalance or generator over-limit output. Figure 13 shows that in the initial stages of training, the reward fluctuates and is mostly negative. Figure 14 presents that at the end of training, the reward is stable at a relatively high level, which means that most of the flows are adjusted to convergence. A few negative rewards are due to the randomness introduced in Section V.

4) Power flow generation

As mentioned in the above sections, the uncertainty factors of the actions and states in the power flow adjustment model can be used to generate convergent power flow samples. Specifically, samples can be generated by executing the same action sequence, or adjustment strategy, learned by the model multiple times. Due to the uncertainty factors introduced into the model, different applications of the same strategy will have slightly different effects on the controlled objects, thus leading to different system states and power flows. As the learning process advances, the action sequence selected in each iteration gradually becomes shorter until it remains nearly stable. Then, the action sequence generated in each iteration is recorded as a strategy, and multiple power flow samples can be obtained by executing the same strategy many times. The proportion of convergent power flow samples among the samples generated in this way is shown in Fig. 15 and Fig. 16.

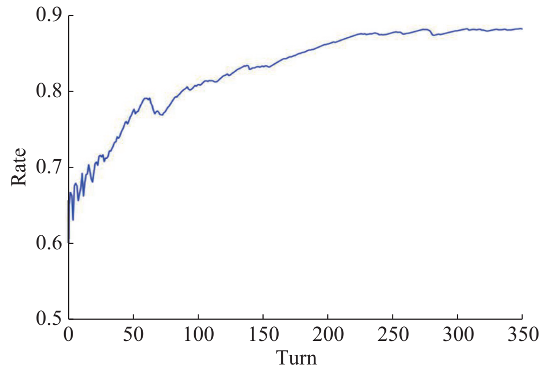

Fig. 15. Variation in convergence proportion for flow sample 1.

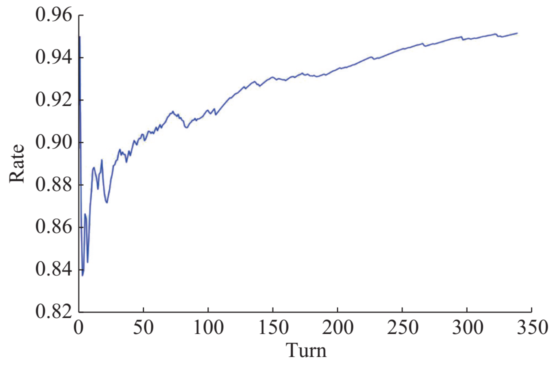

Fig. 16. Variation in convergence proportion for flow sample 2.

It can be observed from the above figures that throughout the training process, the proportion of the convergent power flow samples obtained using the learned strategies increases. As seen in Fig. 15, executing the strategy learned at the 50th iteration yields approximately 72% convergent samples, while the strategy learned at the 300th iteration yields nearly 90% convergent samples. For power flow sample 1, 6288 convergent samples and 840 nonconvergent samples are generated using the strategy learned by the adjustment model, corresponding to a convergence proportion of 88.2%. For power flow sample 2, 6480 convergent samples and 330 nonconvergent samples are obtained, corresponding to a convergence proportion of 95.2%. Compared with the 16.3% convergence proportion of random generation, the efficiency of convergent sample generation can be significantly improved by generating power flow samples based on the reinforcement learning model.

B. Case of IEEE 118-bus System

In this case, the proposed method is verified based on IEEE 118-bus system. The system includes 54 generators, 177 lines and 91 loads. Based on the basic flow, flow samples are generated by adjusting the scale factors of generators and loads with some randomness. A total of 40000 samples are generated, including 15428 convergent samples and 24572 nonconvergent samples. The proportion of convergent power flow samples is 38.57%, which is slightly higher than the case 1 due to the different way of sample generation. The model training time is 8.2 min.

The training and testing process are similar to the first case. In the training process, 80% of the samples are selected as the training set, and the remainder are used as testing samples. The DNN consists of one input layer, four hidden layers and one output layer. The numbers of neurons in each layer are 36, 16, 16, 8, 8, and 1, respectively, which are the same as case 1. After training the discriminator, one sample selected from the nonconvergent samples is used to verify the method. The specific data of the flow is shown in Appendix A Table AII. According to the capacity of the generator, Appendix A lists only some of the key control objects. For the nonconvergent power flow sample, the number of training iterations is set to 2500, and the number of actions required in each iteration of convergence adjustment is recorded. Figure 17 shows the total training process, and Fig. 18 shows the last 200 iterations of the training process.

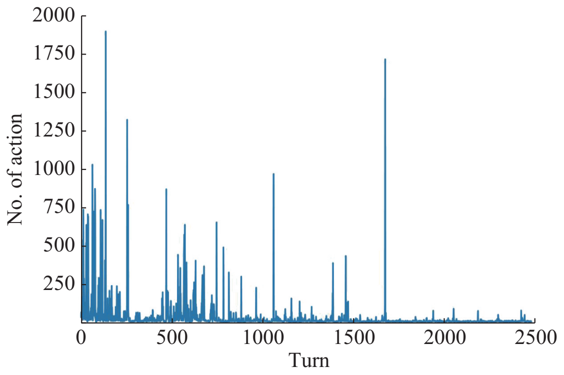

Fig. 17. Iterations for power flow sample of total training process.

It can be seen from the above figures that during the initial stage of training for the power flow adjustment model, the numbers of actions required to achieve convergence are very large and varied greatly. As seen in Fig. 17, the number of actions at the 164th iteration is 209, and the 641st iteration is 405. During the final stage of the training process, the number of actions remained relatively stable at approximately 10, indicating that the state-action value function has converged and an appropriate adjustment strategy has been learned. The minor fluctuation of the numbers is attributed to the uncertainty introduced into the model. Table IV presents the action sequences selected at the 2494th, 2495th, and 2496th iterations. The adjustment policies in these iterations focus on actions 16 and 23, which lead to the adjustment of reactive power of the generators on buses 103 and 25. The reasons for the differences in these action sequences are: ① the adoption of ε-greedy method; ② the state-action function values of different actions may be the same. Figure 19 shows the proportion of convergent power flow samples obtained using the learned strategies.

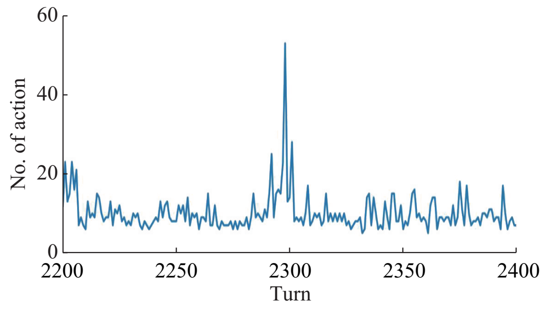

Fig. 18. Last 200 iterations for power flow sample.

TABLE IV
ACTION SEQUENCES SELECTED IN LAST THREE ITERATIONS

Iteration	Action No.
2494	16, 16, 16, 24, 23, 23, 8, 16, 23
2495	16, 16, 7, 23, 23, 5
2496	16, 16, 8, 23, 23, 23, 9

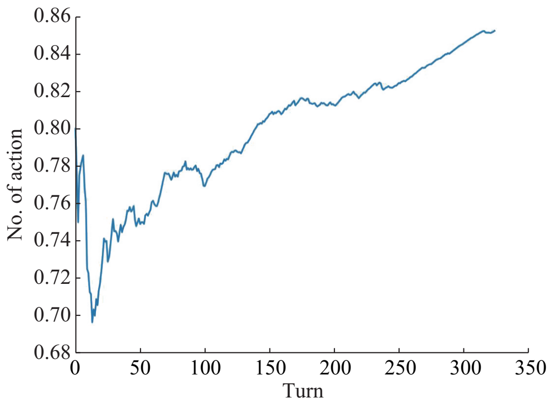

Fig. 19. Rate of convergent flow of first 300 iterations for power flow sample.

Figures 20 and 21 show the number of actions and average reward of each turn during the whole training process. It can be seen that at the beginning stage of the training, the curve fluctuates violently, while at the final stage the curve becomes smooth, which means the model converges. Figures 22 and 23 show the average reward during the beginning and final stages of training process.

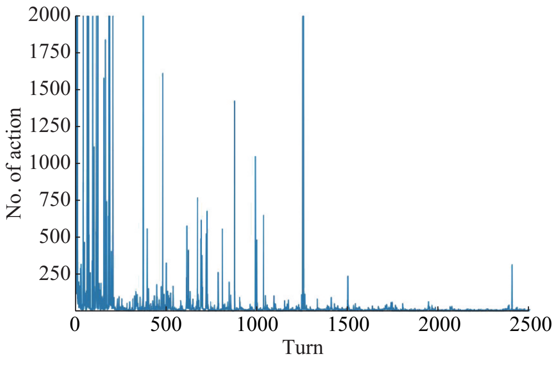

Fig. 20. Number of actions of each turn during whole training process.

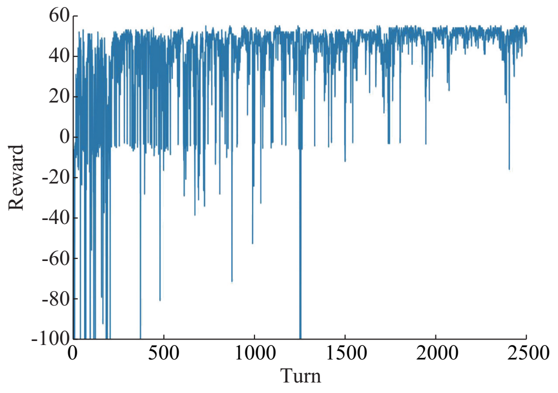

Fig. 21. Average reward for each turn during whole training process.

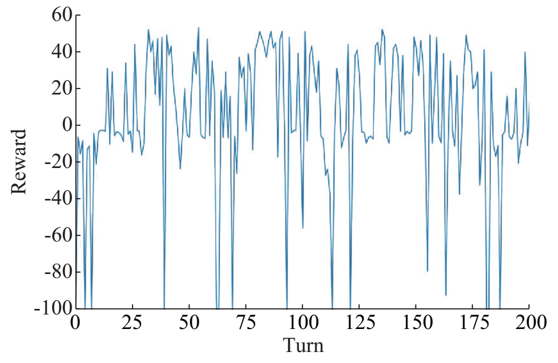

Fig. 22. Average reward for first 200 iterations.

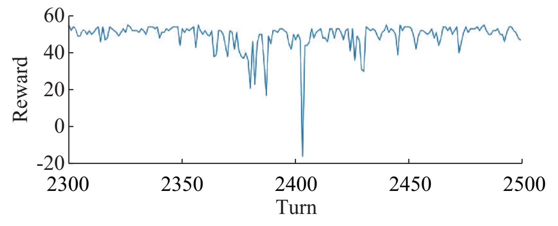

Fig. 23. Average reward for last 200 iterations.

V. CONCLUSION

This paper proposes an intelligent power flow adjustment method based on a deep network and reinforcement learning, and presents an intelligent adjustment framework comprising power flow convergence judgment, power flow convergence adjustment, and power flow generation. First, a DNN is built to judge the convergence of power flow scenarios. Then, intelligent adjustment is realized by using the SARSA algorithm. Finally, an intelligent method of generating convergent power flow samples is realized. The intelligence and effectiveness of the method have been verified in a case study. The following conclusions can be drawn.

① A three-part framework is proposed to address power flow convergence, automatic power flow adjustment, and sample generation. AI method is used to solve the problem to satisfy the actual demand of power system engineering. ② The power flow convergence discriminator can determine the convergence of power flows much faster than commercial software based on specified initial settings. It also provides more useful information for training the adjustment model. ③ The intelligent adjustment model can adjust a nonconvergent power flow to a convergent one without any manual intervention. ④ The learned adjustment strategy can be used to generate a large number of different convergent power flow samples, with much higher efficiency than that of random generation.

In future research, the operation conditions and fault checks should also be considered in the adjustment model when generating power flow samples, which improves the rationality of power flow generation and makes the results more useful for practical operation.

Appendix A

Figure A1 shows the three parts of the framework. Tables AI and AII show the specific power flow data of samples in cases 1 and 2. Table AII lists only some critical features of the flow sample.

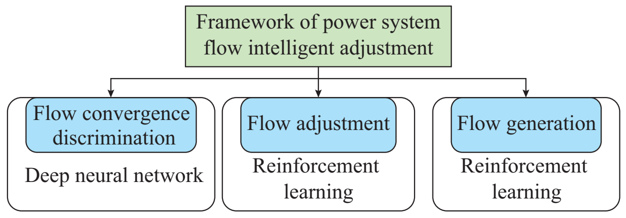

Fig. A1. Three parts of whole framework.

TABLE AI
Flow Data of CEPRI 36-bus System

System state	Flow sample 1	Flow sample 2	System state	Flow sample 1	Flow sample 2
BUS2_gen_p	5.22	4.96	306_load_p	1.07	3.31
BUS3_gen_p	4.10	4.37	307_load_p	1.86	1.95
BUS4_gen_p	3.62	4.54	308_load_p	1.96	2.68
BUS5_gen_p	2.41	4.46	310_load_p	0.89	0.38
BUS6_gen_p	0.19	5.29	300_load_p	5.83	2.45
BUS7_gen_p	4.48	4.26	301_load_q	1.94	0.73
BUS8_gen_p	0.68	1.36	302_load_q	0.11	1.85
BUS2_gen_q	0.65	1.92	303_load_q	0.67	0.12
BUS4_gen_q	0.48	1.26	304_load_q	0.26	1.04
BUS5_gen_q	0.04	1.36	305_load_q	0.44	2.86
301_load_p	2.43	4.63	306_load_q	2.82	2.70
302_load_p	5.89	5.86	307_load_q	2.50	0.77
303_load_p	1.17	3.03	308_load_q	1.85	0.90
304_load_p	3.64	0.15	310_load_q	1.46	1.19
305_load_p	0.03	5.76	300_load_q	0.02	1.09

TABLE AII
Flow Data of IEEE 118-bus System

System state	Flow sample	System state	Flow sample	System state	Flow sample	System state	Flow sample
BUS1_gen_p	1.00	BUS1_gen_q	0.00	BUS1_load_p	0.51	BUS1_load_q	0.27
BUS2_gen_p	4.50	BUS2_gen_q	2.00	BUS100_load_p	0.20	BUS100_load_q	0.09
BUS100_gen_p	2.52	BUS100_gen_q	1.55	BUS101_load_p	0.37	BUS101_load_p	0.18
BUS103_gen_p	0.40	BUS103_gen_q	4.46	BUS102_load_p	0.23	BUS102_load_q	0.16
BUS104_gen_p	0.19	BUS104_gen_q	0.40	BUS103_load_p	0.38	BUS103_load_q	0.25
BUS107_gen_p	–0.22	BUS107_gen_q	0.00	BUS104_load_p	0.28	BUS104_load_q	0.12
BUS111_gen_p	0.36	BUS111_gen_q	10.00	BUS105_load_p	0.31	BUS105_load_q	0.26
BUS112_gen_p	–0.43	BUS112_gen_q	0.00	BUS106_load_p	0.43	BUS106_load_q	0.16
BUS113_gen_p	1.00	BUS113_gen_q	0.00	BUS107_load_p	0.28	BUS107_load_q	0.12
BUS116_gen_p	–1.84	BUS116_gen_q	1.00	BUS108_load_p	0.02	BUS108_load_q	0.01
BUS25_gen_p	2.20	BUS25_gen_q	1.40	BUS109_load_p	0.08	BUS109_load_q	0.03
BUS26_gen_p	3.14	BUS26_gen_q	10.00	BUS11_load_p	0.70	BUS11_load_q	0.23
BUS49_gen_p	2.04	BUS49_gen_q	2.10	BUS110_load_p	0.39	BUS110_load_q	0.30

References

[1] S. Zhang, Y.Chen, F. Li *et al.*, “Functional design and implementation of collaborative system for power grid operation mode calculation,” *Power System Technology*, vol. 36, no. 10, pp. 270-274, Oct. 2012.

[2] Z. Du, R. Zhang, Y. Wang, *et al.*, “A constrained load flow method to obtain power system operation mode,” *Proceedings of the CSEE*, vol. 35, no. 4, pp. 840-847, Apr. 2015.

[3] Z. Yan, X. Fan, W. Zhao *et al.*, “Improving the convergence of power flow calculation by a self-adaptive levenberg-marquardt method,” *Proceedings of the CSEE*, vol. 35, no. 8, pp. 1909-1918, Aug. 2015.

[4] Y. Tamura, Y. Nakanishi, and S. Iwamoto, “On the multiple solution structure, singular point and existence condition of the multiple load-flow solutions,” *IEEE Transactions on Power Systems*, vol. 99, no. 4, pp. 1322-1322, Feb. 1980.

[5] J. T. Overbye, “A power flow measure for unsolvable cases,” *IEEE Transactions on Power Systems*, vol. 9, no. 3, pp. 1359-1365, Aug. 1994.

[6] T. V. Custsem, “An approach to corrective control of voltage instability using simulation and sensitivity,” *IEEE Transactions on Power Systems*, vol. 10, no. 2, pp. 616-622, May 1995.

[7] L. V. Barboza, A. A. P. Lerm, and R. Salgado, “Unsolvable power Flow – restoring solutions of the electric power network equations,” in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, Jul. 2005, pp. 5294-5297.

[8] M. Li, J. Chen, H. Chen *et al.*, “Load flow regulation for unsolvable cases in a power system,” *Automation of Electric Power Systems.*, vol. 30, no. 8, pp. 11-15, Aug. 2006.

[9] S. Granville. J. C. O. Mello, and A. C. G. Melo, “Application of interior point methods to power flow unsolvability,” *IEEE Transactions on Power Systems*, vol. 11, no. 2, pp. 1096-1103, May 1996.

[10] F. Hong, “Research on the power flow automatic adjustment methods in power system,” Ph.D. dissertation, Department of Electrical Engineering, Huazhong University of Science and Technology, Wuhan, China, 2011.

[11] Y. Lin, H. Sun, W. Wu *et al.*, “A schedule power flow auto generating technology in day-ahead security validation,” *Automation of Electric Power Systems*, vol. 36, no. 20, pp. 68-73, Oct. 2012.

[12] Y. Wang, J. Hou, S. Ma *et al.*, “A method of automatic integration and regulation of power flow data for security and stability check of generation scheduling analysis,” *Power System Technology*, vol. 34, no. 4, pp. 100-104, Apr. 2010.

[13] D. Zhang, X. Han, and C. Deng, “Review on the research and practice of deep learning and reinforcement learning in smart grids,” *CSEE Journal of Power and Energy Systems*, vol. 4, no. 3, pp. 362-370, Sept. 2018.

[14] O. Russakovsky, J. Deng, H. Su *et al.*, “Image net large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, Apr. 2015.

[15] A. Graves, A. R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, New York, USA, Apr. 1998, pp. 1-5.

[16] K. Cho, B. V. Merrienboer, C. Gulcehre *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *Proceedings of 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1724-1733.

[17] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529-533, Feb. 2015.

[18] D. Silver, A. Huang, C. J. Maddison *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484-489, Jan. 2016.

[19] N. Zhou, J. Liao, Q. Wang *et al.*, “Analysis and prospect of deep learning application in smart grid,” *Automation of Electric Power Systems*, vol. 43, no. 4, pp. 180-191, Feb. 2019.

[20] T. Ouyang, Y. He, H. Li *et al.*, “Modeling and forecasting short-term power load with copula model and deep belief network,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 2, pp. 127-136, Apr. 2019.

[21] D. L. Marino, K. Amarasinghe, and M. Manic, “Building energy load forecasting using deep neural networks,” in *Proceedings of Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Florence, Italy, Oct. 2016, pp. 1-6.

[22] W. Liu, D. Zhang, X. Wang *et al.*, “A decision making strategy for generating unit tripping under emergency circumstances based on deep reinforcement learning,” *Proceedings of the CSEE*, vol. 38, no. 1, pp. 109-119, Jan. 2018.

[23] T. Yu, B. Zhou, K. Chan *et al.*, “*Q*-learning based dynamic optimal CPS control methodology for interconnected power systems,” *Proceedings of the CSEE*, vol. 29, no. 19, pp. 13-19, Jul. 2009.

[24] E. A. Jasmin, T. P. I. Ahamed, and V. P. J. Raj, “Reinforcement learning approaches to economic dispatch problem,” *International Journal of Electrical Power & Energy Systems*, vol. 33, no. 4, pp. 836-845, May 2011.

[25] M. Chu, X. Liao, H. Li *et al.*, “Power control in energy harvesting multiple access system with reinforcement learning,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9175-9186, Jul. 2019.

[26] J. Gong and Y. Liu. “Coordinated optimization of active distribution network based on deep deterministic policy gradient algorithm,” *Automation of Electric Power systems*, vol. 44, no. 6, pp. 114-120, Mar. 2020.

[27] H. Xu, Z. Yu, Q. Zheng *et al.*, “Deep reinforcement learning-based tie-line power adjustment method for power system operation state calculation,” *IEEE Access*, vol. 7, pp.156160-156174, Oct. 2019.

[28] L. Zheng, W. Hu, Y. Zhou *et al.*, “Deep belief network based nonlinear representation learning for transient stability assessment,” in *Proceedings of IEEE PES General Meeting*, Chicago, USA, Jul. 2017, pp. 1-5.

Shuang Wu received his B.S. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2016, and he is now a Ph.D. candidate at Tsinghua University, Beijing, China. His research interests include power system analysis and control as well as applications of big data and artificial intelligence technology in power systems.

Wei Hu received his B.S. and Ph.D. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1998 and 2002, respectively, and he is now an Associate Professor with Tsinghua University. His research interests include power system analysis and control, applications of big data technology in power systems, multitype power generator-grid coordination and control, and optimized control of renewable energy and energy storage systems.

Zongxiang Lu received his B.S. and Ph.D. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1998 and 2002, respectively. Since 2002, he has been with Tsinghua University, Beijing, China, where he is currently an Associate Professor of electrical engineering. He is the Fellow of IET, and the Senior Member of IEEE and CSEE. His research interests include large-scale wind power/PV station integration analysis and control, energy and electricity strategy planning, power system reliability, DG, and microgrid.

Yujia Gu received his master’s degree in electrical engineering from China Agricultural University, Beijing, China in July 2011. From August 2011 to the present, he has been engaged in power system analysis and simulation at the Power Research Institute of State Grid Ningxia Electric Power Co., Ltd., Yinchuan, China. His research interests include power system analysis and control.

Bei Tian received her master’s degree from Northeast Electric Power University, Jilin, China, in 2000. Since July 2000, she has worked at the Power Research Institute of State Grid Ningxia Electric Power Co., Ltd., Yinchuan, China, and has been engaged in power system simulation analysis and various scientific and technological innovations. Her research interests include power system operation and control.

Hongqiang Li received his master’s degree in electrical engineering from Southwest Jiaotong University, Chengdu, China, in 2014. From August 2014 to the present, he has been working in Power Research Institute of State Grid Ningxia Electric Power Co., Ltd., Yinchuan, China. His research interests include power system analysis and simulation.