Game Account Valuation Platform Based on Deep Reinforcement Learning Algorithm

1 Demand Analysis

With the rapid development of Internet technology and computer technology, there are various kinds of games in the market. With the increasing demand of contemporary people for entertainment and social activities, the game industry has a vast market. Players have to put in time and effort to get a good gaming experience, and it is obviously unrealistic to pay a lot of time and effort for account upgrading in the fast-paced modern life, so game account trading has a wide market space. However, the game types are endless, and the valuation of game accounts is still a blank field that still needs to be explored.

The main purpose of this document is to conduct a comprehensive requirements analysis of a web page based on the Fifth Personality game in order to clarify the requirements in terms of functionality, performance and constraints needed for the web page. Specifically, this document will detail the requirements for the processing of user input data, the design principles and implementation of the valuation algorithm, and the design and implementation of the user interface. By analyzing and planning the requirements in these areas, we will ensure that the developed web pages will meet user requirements, have good user-friendliness and data security, and operate stably and efficiently.

1.1 Functional requirements

Role	Ability
user	The main functions of the platform include the user inputting the number of props in the game as well as the level and other parameters, which are calculated by the background program

a. User input: The user should be able to input the number of props and various parameters he has in the game, including the data of spirits, skins, equipment, etc.

b. Data processing: The system should be able to process and calculate the data input by the user to obtain the account valuation, including the algorithm design for weighted calculation of various props and attributes, market reference, etc.

c. Valuation display: The system should be able to display his account valuation to the user, including information such as the value of the account, the market situation and the recommended selling price in textual or graphical presentation.

1.2 Sample

Use Case Name	Account Valuation
Implementation	account_evaluation

name	
Case description	User requests account valuation from the system through this use case
Participant	User
Pre-conditions	Enter the values of the parameters in the page
Post-condition	Output the corresponding result
Main event flow	User enters the account information according to the actual situation User clicks on the calculate button The backend checks the entered values and performs the calculation If the input value is correct, the final result is output
Alternate event flow	Incorrect type of input value, web page prompts user to re-enter
Business rules	The user can get the final result by entering the parameter values
Entity involved	User

1.3 Non-functional Requirements

a. User-friendliness: The system should have good user-friendliness, including requirements for a concise and clear user interface, easy-to-operate input methods, and accurate and reliable valuation results.

b. Data security: The system should have good data security, including the guarantee of confidentiality, integrity and reliability of user data.

1.4 User Requirements

a. User group: The users of this system are mainly players who like to play the fifth personality game, and they want to understand the market value and competitiveness of their accounts through valuation.

b. Usage scenarios: Users may use this system at some point in the game, for example, when they want to sell their account or to better understand the value of their account.

1.5 System Requirements

Technology platform: This system needs to be implemented based on Web technology, including technical support in HTML, CSS, JavaScript, etc.

2 System Design Based on Flask and Vue

2.1 Base Class Design

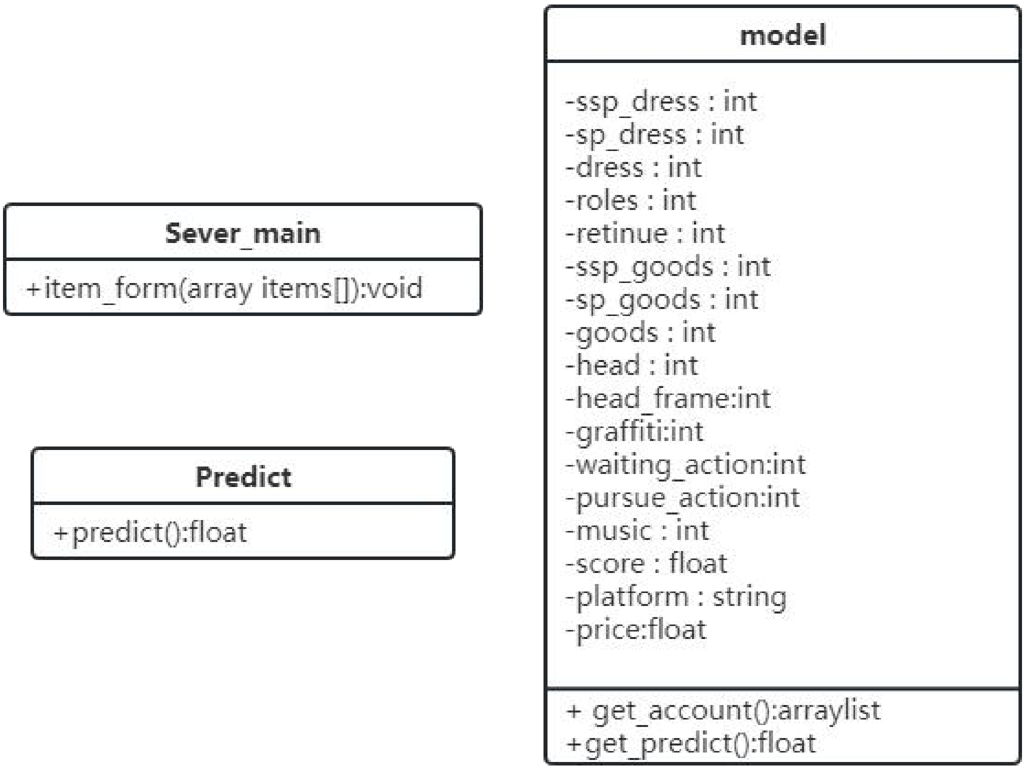

Fig 2.1 Base Class Design

2.2 Page UI Design

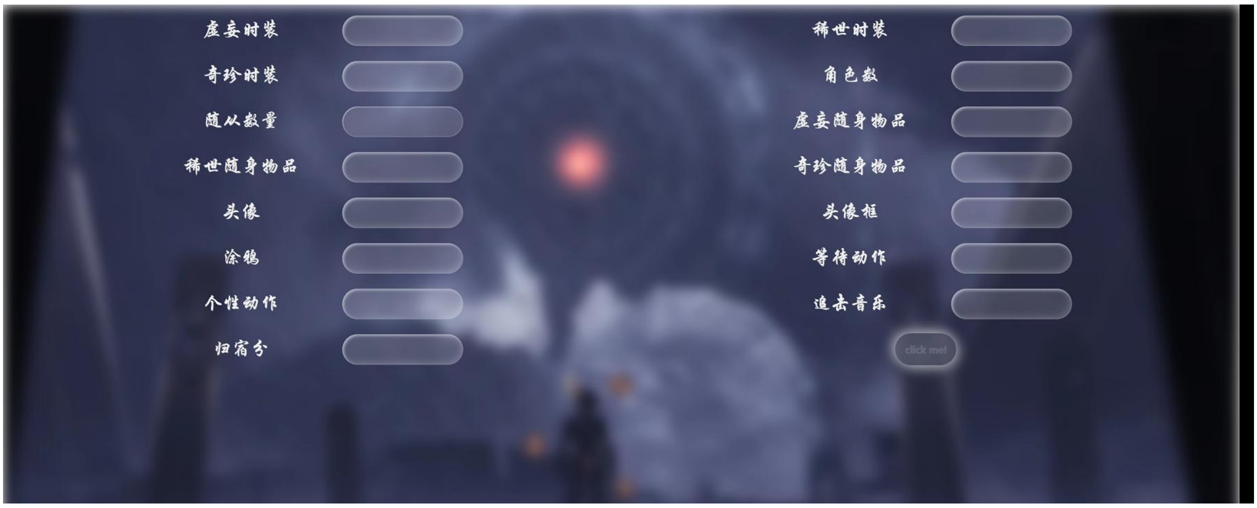

Fig 2.2 Page UI Design

2.3 Business Flow Design

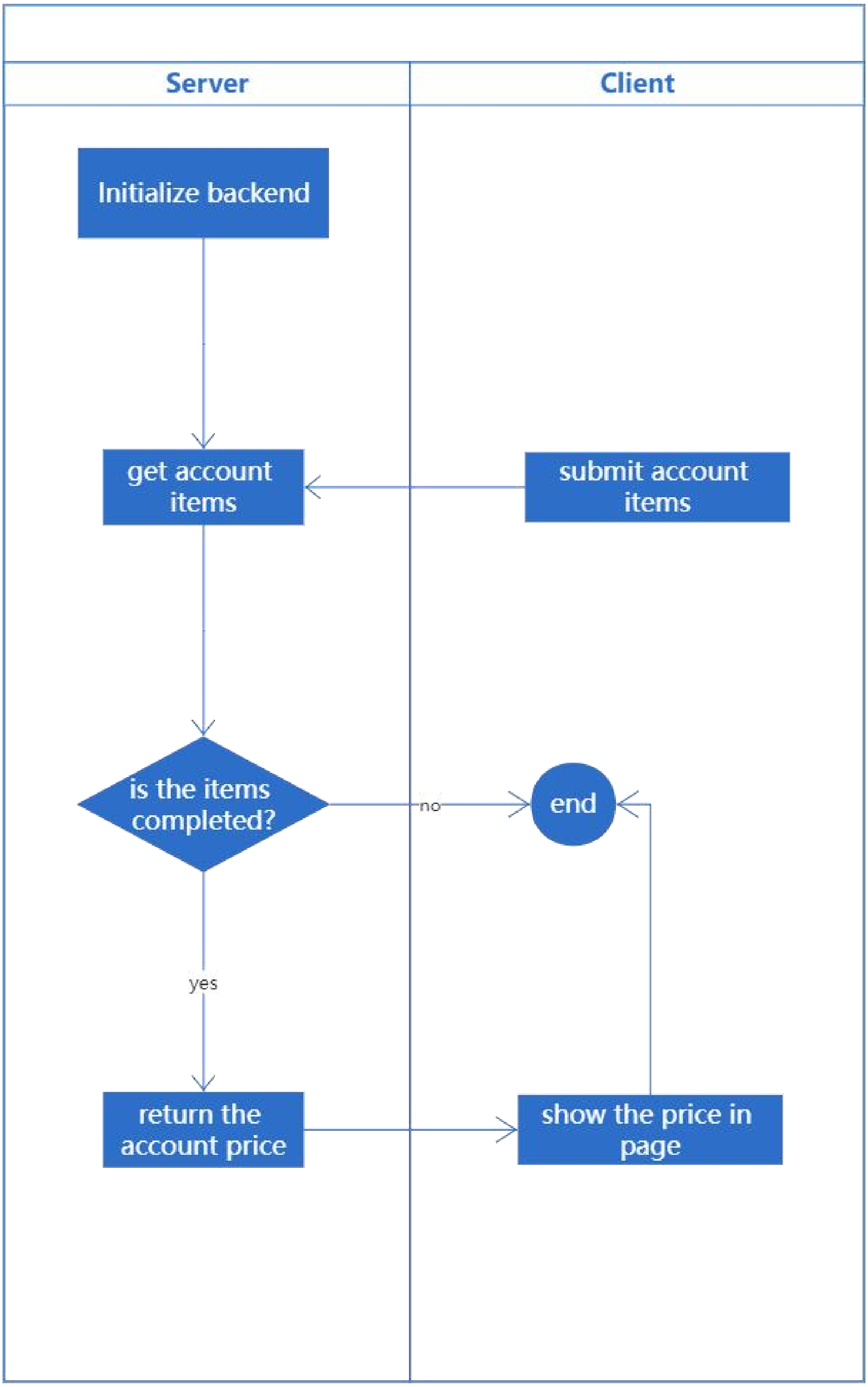

Fig 2.3 Business Flow Design

3 Deep Reinforcement Learning-based Game Account Evaluation Algorithm Design

3.1 Overview

In this section, the DDPG-based game account evaluation algorithm is described in detail. The dataset contains account information of 800 accounts to be traded from January to March 2023 in Trading Cat and Hidden Treasure. The algorithm generates a set of states (State) from the feature values of the data into the neural network to derive the weight γ as the output action, and gets the reward value (Reward) of this set of data by the designed reward function, and then stores this set of states, weights, and reward values into the experience pool (Buffer) for the update of the neural network.

3.2 State Space

The state space includes: {virtual fashion, rare fashion, exotic fashion, number of characters, number of followers, virtual portable items, rare portable items, exotic portable items, avatars, avatar frames, graffiti, waiting actions, personality actions, chase music, homecoming score, trading platform, price} (ssp_dress,sp_dress,dress,roles,retinue.ssp_goods,sp_goods,goods,head,head_frame,graffiti,waiting_actoion,music,socre,platform,price) 17 parameters make up the state space.

3.3 Action space: the predicted price of the output (γ)

3.4 Reward function: the result of the reward function is determined by the size of the deviation of the single-step network output action from the expert guidance, and the maximum reward value is 0.1 for a single step, and the larger the deviation, the smaller the reward value. The reward function is:

$$r = 0.1-| \gamma - \beta |$$

where γ is the normalized value of the output valuation and β is the normalized value of the expert valuation of the item.

3.5 Simulation of Expert Experience Description

We simulate the price given by the game account expert for the account after evaluation with the linear primitive function:

P=state[0]*0.15+state[1]*0.08+state[2]*0.06+state[3]*0.06+state[4] *0.05+state[5]*0.15+state[6]*0.08+state[7]*0.06+state[8]*0.05+ state[9] *0.04+ state[10]*0.06+ state[11]*0.05+ state[12]*0.06+ state[13]*0.04+ state[14]*0.08* state[16]*0.13

3.6 Data Normalization Description

In order to better simulate expert experience and to meet the requirements of model training, the training data are pre-normalized. The data under the same eigenvalue are normalized to dimensionless data and become scalar. The normalization formula used is： $x_{归一化} = \frac{x_{原始} - x_{\min}}{x_{\max} - x_{\min}}$

3.7 Neural Network Description

(a) Network update process:

The output of DDPG algorithm is an action and, we can consider the Actor network in DDPG as a "black cloth", when we input a set of states, this "black cloth" will be When we input a set of states, this "black cloth" will be cut along the position of the states, and along the edge of the "black cloth" is the Q value corresponding to different actions in this state, and the task of the Actor is to keep finding this maximum value through gradient ascending. The input of the Critic network is to predict the Q value, and it has two inputs: Action and State, which need to be input into the Critic network together, and the loss value used is the same as AC, and TD-error is used as the basis for judging the current behavior policy.

(b) Network Structure Diagram:

Actor network structure diagram:

The actor network adopts a full network connection structure of 17-30-30-1 with three layers. Among them, the number of nodes in the input layer is 17, the number of nodes in the hidden layer is 30, and the number of nodes in the output layer is 1. The relu activation function is used in the input and hidden layers, and the sigmoid activation function is used in the output layer to compress the results between [0,1].

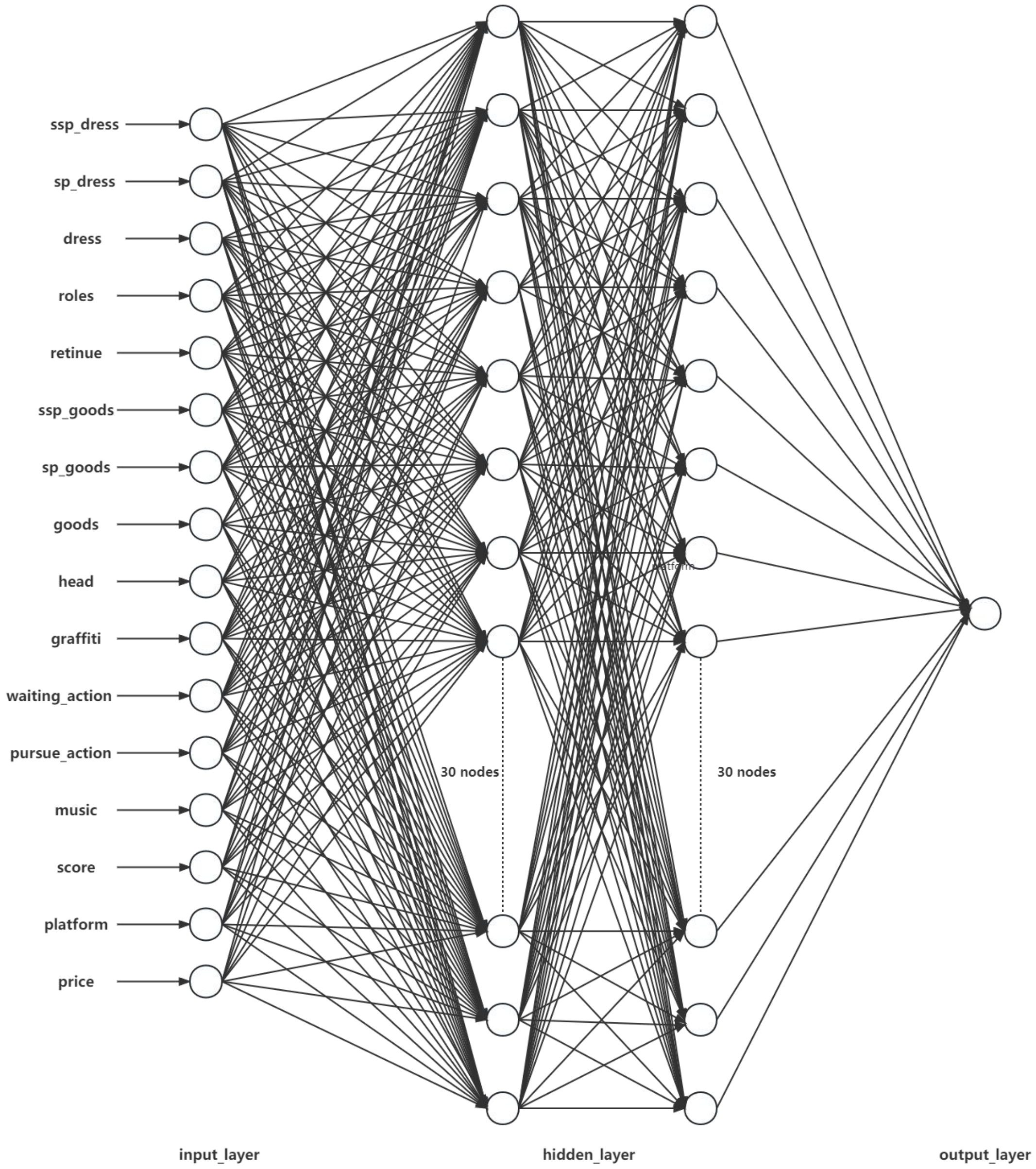

Fig 3.1 Network Structure

critic network structure diagram:

The critic network adopts a full network connection structure of 7-60-60-1. The size of the input layer corresponds to the sum of the size of the state space and the size of the action space, which is 18 nodes, the hidden layer has 60 nodes, and the output layer has 1 node. Among them, the hidden layer uses the relu activation function.

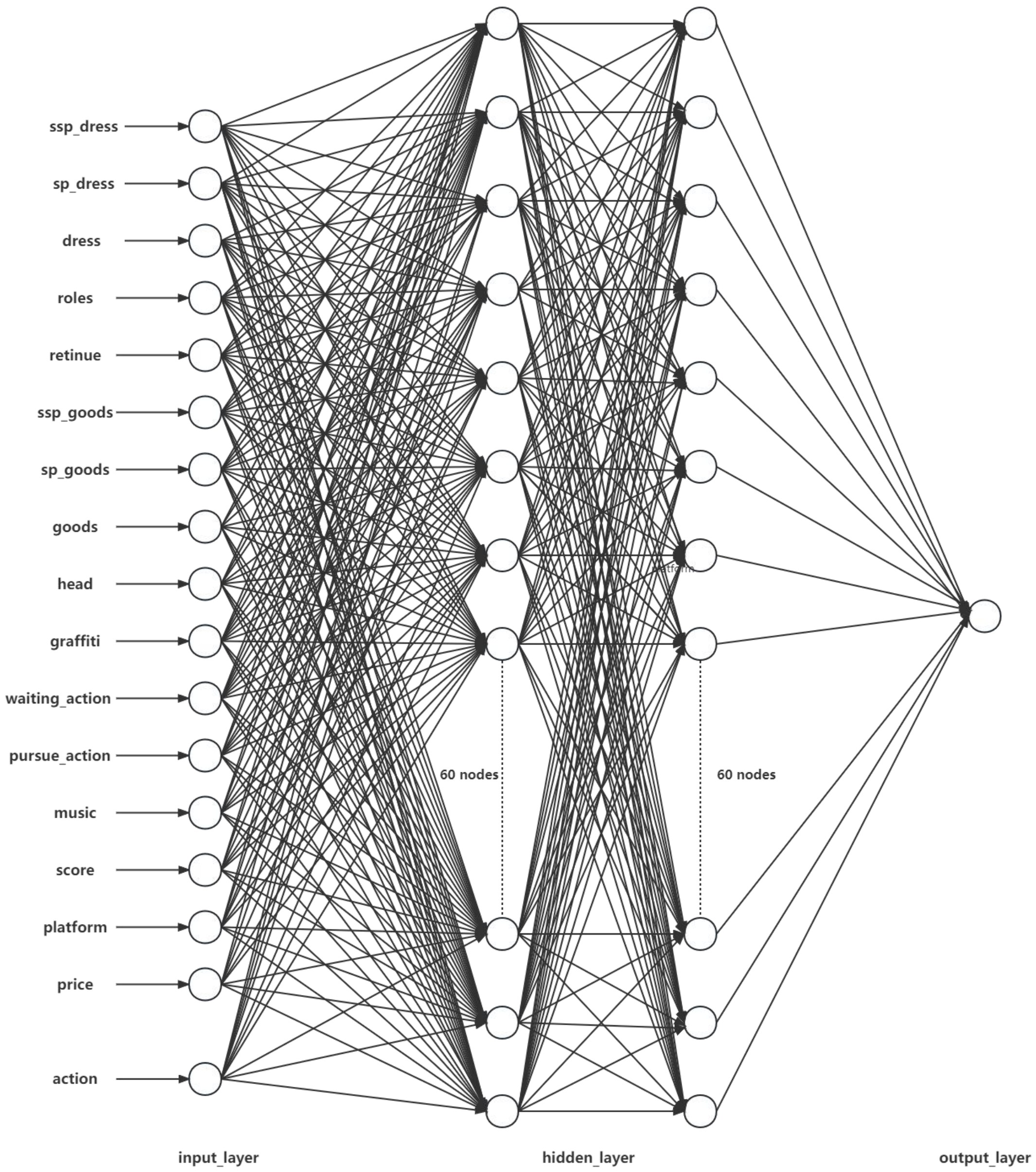

Fig3.2 Critic Network

3.8 Detailed description of the training algorithm: lines 1 to 5 describe the algorithm initialization parameters, lines 6 to 18 describe the process by which the algorithm obtains the current action and state, and lines 19 to 29 describe the process by which the algorithm selects samples from the buffer each time and updates the parameters using a gradient ascent.

Algorithm 3.1 DDPG training algorithm

```
Input: Actor network and Critic network
       Buffer size buffer_size
       Batch size batch_size
       Number of iterations used in each update sample_update_times
```

Output: trained Actor-Critic model

Algorithm steps:

```
Initialize net_actor;//Initialize Actor network
Initialize net_critic;//Initialize Critic network
Initialize buffer←get_buffer(buffer_size);//Initialize buffer
Initialize Rlist ← [ ];//Initialize long-term cumulative returns at each time step R list
Initialize i← 0;//Initialize the current number of training steps is 0
while receive a sample ε from the sampling node do//Receive training samples
    buffer.add_tail(ε);//Storage samples
    if buffer.size > buffer_size:
      i←i+ 1;
        Initialize R_temp← 0;
      foreach ε in getReversList(buffer) do
          if getIndex(buffer,ε) = buffer.size() − 1 do
              s ←ε.getState();
              V← net_critic.getValueFunc(s);
              R_temp←V+r*R_temp;
          else
              R_temp←getReward(ε)+r*R_temp;
              Initialize j← 0;//Number of iterations
              while j < step do
                  foreach ε in sampleSubBuffer(buffer,batch_size) do //Multiple iterations
                      s ← ε.getState();
                      a ← a.getActoin();
                      index← getIndex(buffer, ε);
                      R ← Rlist. getValueByIndex(index);
                      V← net_critic_getValueByFunc(s);
                      A← R-V;
                      actor_loss = getActorLoss(A,r(Θ),ε);//based on equation (5.8)
                      ctitic_loss = getCriticLoss(A);// based on equation (5.9)
                      updateActorNet(actor_loss);//Update Actor Network
                      updateCriticNet(critic_loss);//Update Critic Network
                  end for
                j ← j+ 1;
      end if
    end
  end for
end
```

3.9 Experiments and Experimental Analysis

3.9.1 Hyperparameter Selection and Analysis

In order to explore the appropriate hyperparameters to enhance the network efficiency, the Actor network learning rate of 0.01, Critic network learning rate of 0.001, and batch size of 32 were selected; the Actor network learning rate of 0.01, Critic network learning rate of 0.001, and batch size of 64; the Actor network learning rate of 0.001, Critic network learning rate of 0.0001 and a batch size of 32 as a control.

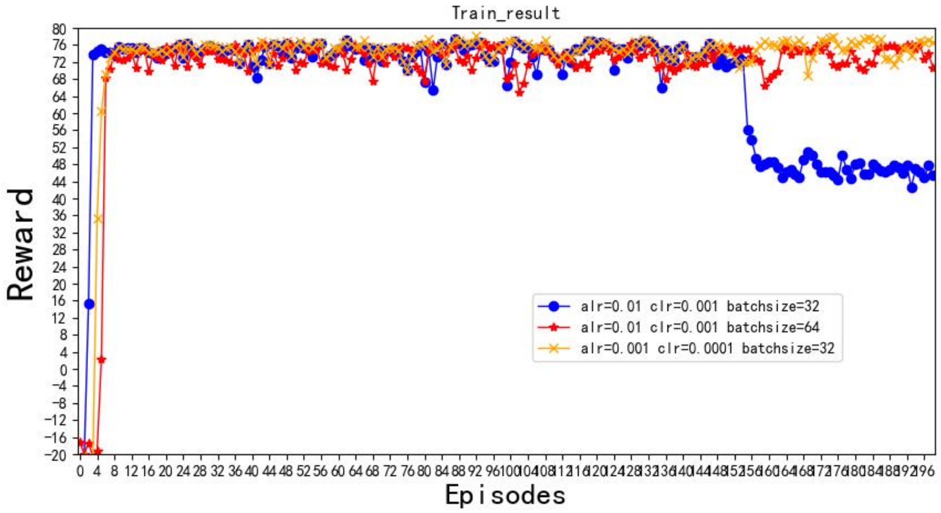

Fig 3.3 Experimental Analysis

Analysis of results: In the total number of experiments with 200 cycles, it is not difficult to find that the DDPG algorithm is able to converge the neural network quickly. Analyzing the experimental comparison results, the network is trained most efficiently when the learning rate of Actor network is 0.001, the learning rate of

Critic network is 0.0001, and the batch size is 32. Therefore, we can select the network under this group of hyperparameters as the network for in-depth testing.

3.9.2 Analysis of test results

We select 50 account data as the test set. Figure 3.4 shows the comparison images of the expert predicted account price and the neural network output price. Figure 3.5 shows the single-step output reward value on the test set, as the closer the reward is to 0.1, the smaller the deviation value between the expert prediction and the network output.

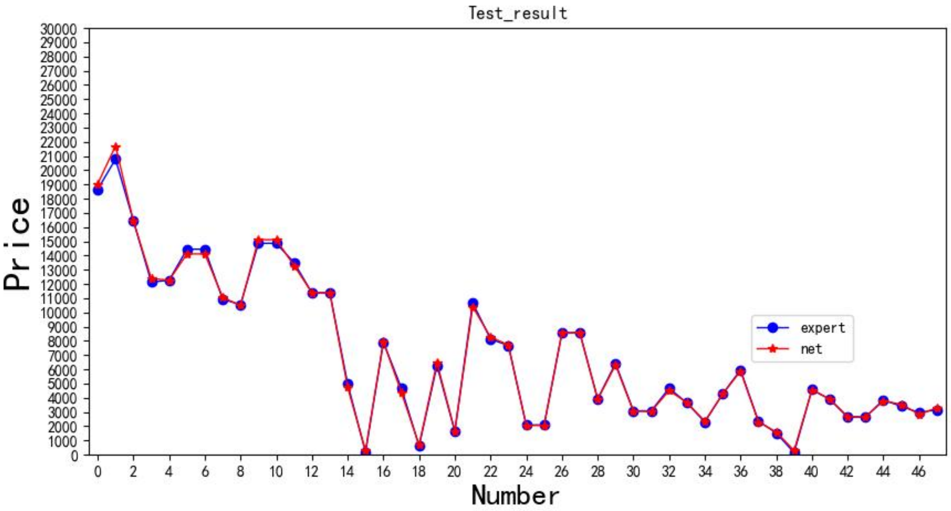

Fig 3.4 Price Chart

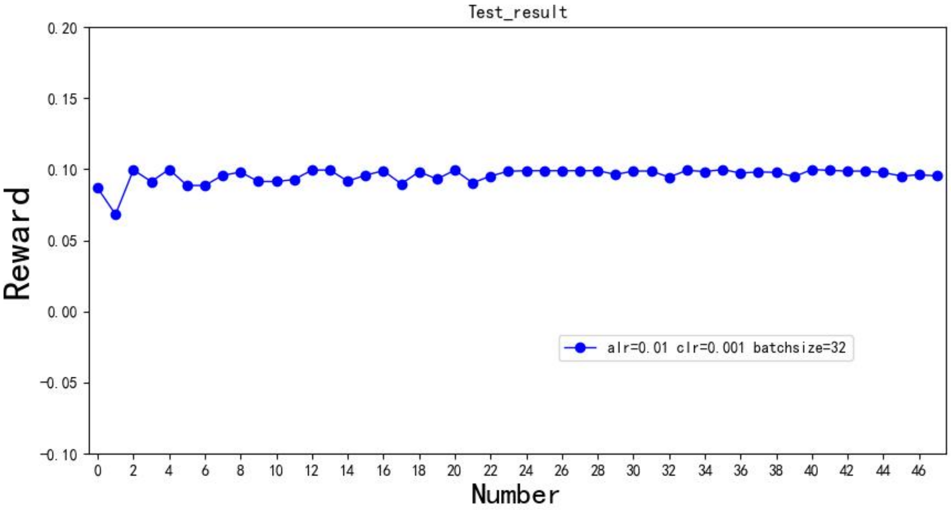

Fig3.5 Reward Chart

Result analysis: The test results reflect that the network has a very good fit for the expert experience and the neural network is able to give an accurate predicted price for the account.