Research on Dynamic Big Data Processing and Optimization Model Based on Optimized PSO and Deep Reinforcement Learning

Zhenwei Tang[1], Chuanjian Jiang[2*], Xi Zhang[3]
Corresponding Email: ChuanjianJiang@outlook.com
[1]Chongqing University of Chinese Medicine, Chongqing 402760, China
[2]Chongqing College of International Business and Economics, Chongqing 401520, China
[3]Chongqing University of Chinese Medicine, Chongqing 402760, China

Keywords: big data processing, optimize PSO, deep reinforcement learning, dynamic model

Received: March 15, 2025

In the era of big data, the optimal processing of dynamically better data has become the core concern of academia and industry, and traditional methods are often difficult to meet the high standards of real-time and accuracy due to the complexity, variability, high-speed liquidity, and large scale of data. To this end, this study proposes a dynamic big data processing and optimization model, which improves the particle swarm optimization algorithm (PSO) by introducing adaptive weights and dynamic learning coefficients to improve the global exploration ability and convergence speed, and integrates a hybrid framework of deep reinforcement learning (DRL) (combined with policy gradients such as proximal policy optimization PPO and Q learning) to achieve big data optimization by using its feature extraction and policy adjustment capabilities. Based on real datasets such as 1.2 million pieces of financial transaction data and 8.5 million pieces of social media travel data, the results show that compared with traditional methods (such as traditional PSO, DQN, and independent LSTM-Transformer models), the new model has a 35% increase in data processing speed, a 20% increase in the accuracy of classification tasks (F1 score: 0.92 vs. 0.76 of DQN), a 40% increase in the real-time response ability of dynamic data streams, and a 25% increase in computing resource utilization efficiency. The study elaborates on model architecture innovation, dataset source and scale, benchmarking methods, and key performance indicators (such as processing speed, accuracy, real-time response, and resource efficiency), and provides efficient and scalable solutions for scenarios such as financial risk management and real-time recommendation systems.

Povzetek: Razvit je model za obdelavo in optimizacijo dinamičnih velikih podatkov, ki uporablja izboljšani PSO in globoko ojačevalno učenje za povečanje hitrosti obdelave, točnosti in odzivnosti v realnem času.

1 Introduction

With the development of information technology, the era of big data has arrived, and various industries are facing the test of massive data processing and analysis [1, 2]. Dynamic big data processing requires efficient processing speed and real-time response to data changes to adapt to business needs [3]. Old data processing methods often fail to meet the standards, so exploring new dynamic big data processing and optimization models is necessary.

Among many optimization algorithms, particle swarm optimization (PSO) has attracted widespread attention because of its unique swarm intelligence and fast convergence [4, 5]. PSO has strong global search ability by simulating the foraging behavior of birds and using the historical information of individuals and groups to guide the search [6]. However, PSO is prone to local optimal solutions for complex problems, and the algorithm's performance is bitterly affected by parameter selection [7]. Therefore, researchers have proposed various improved PSO methods, such as adaptive weight adjustment and dynamic learning factors, to improve the performance and adaptability of PSO.

As another important branch in artificial intelligence, deep reinforcement learning shows strong potential in complex tasks by combining the representation learning of deep learning with the decision-making ability of reinforcement learning [8, 9]. Deep reinforcement learning models' environmental states through neural networks and optimizes decision-making strategies through trial-and-error learning, suitable for dynamically changing environments [10]. It is pointed out that deep reinforcement learning has achieved remarkable results in resource management, path planning, and other fields, providing new ideas for dynamic big data processing [11]. Combining PSO with deep reinforcement learning and taking advantage of both advantages is expected to achieve high efficiency and robustness in dynamic big data processing and optimization models. PSO can be used to optimize network parameters in deep reinforcement learning and improve the convergence speed and performance of the model. However, deep reinforcement learning can provide richer search

information for PSO and guide PSO to escape local optimum [12, 13]. While theoretically sound, this fusion method requires empirical validation in practical scenarios.

Dynamic big data processing includes data acquisition, storage, processing, analysis, and other links, each facing different challenges [14]. Data acquisition requires real-time and accuracy, storage requires efficiency and scalability, processing requires rapidity and flexibility, and analysis requires accuracy and interpretability. Traditional data processing methods are often optimized for a particular link, lacking holistic consideration [15]. The dynamic big data processing and optimization model based on optimized PSO and deep reinforcement learning can achieve collaborative optimization of each link and improve the overall processing efficiency.

Dynamic big data processing and optimization models face many difficulties in practical application scenarios. First, the amount of data is expanding rapidly, and the requirements for model calculation and storage performance are more stringent. Secondly, the variety and complexity of data increases the challenge of model design. In addition, the demand for real-time requires the model to have the ability to respond quickly and adjust in time. In response to these challenges, this study will explore the specific applications of optimized PSO and deep reinforcement learning in dynamic big data processing and propose corresponding solutions. The research shows that PSO has better advantages in optimizing the structure of neural networks and can betterly improve the performance of neural networks. The decision-making ability of deep reinforcement learning in a dynamic environment has also been widely recognized. This study will further explore the fusion mechanism of PSO and deep reinforcement learning to achieve high efficiency and robustness of dynamic big data processing and optimization models.

This study aims to explore the dynamic big data processing and optimization model based on optimized PSO and deep reinforcement learning. Through theoretical analysis and experimental verification, it reveals its advantages in improving processing efficiency and enhancing model adaptability, providing new theoretical support and practical guidance for dynamic big data processing, and promoting the development and application of big data technology.

2 Theoretical basis and key technologies

2.1 Particle swarm optimization (PSO) principle

In recent years, many researchers have used swarm intelligence optimization algorithms to improve the neural network training process. The swarm intelligence optimization algorithm has higher global convergence and robustness and does not rely on problem feature information. It can effectively exert the generalization mapping ability of neural networks, betterly improving convergence efficiency and strengthening learning ability [16, 17]. Among the swarm intelligence optimization algorithms, the PSO algorithm has higher potential in neural network optimization with the help of simple and efficient random exploration methods [18].

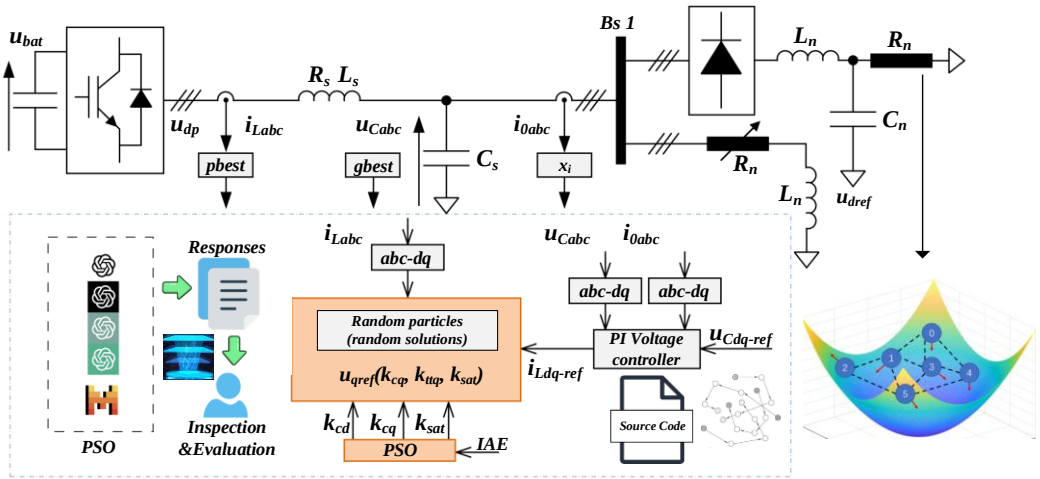

Figure 1: PSO improved neural network framework

The PSO-improved neural network framework is detailed in Figure 1. The algorithm's core is to use PSO instead of traditional training means (such as BP) to optimize NN parameters [19]. The particle symbolizes the parameter vector, and seeking the global optimal value is the step of obtaining the optimal parameter [20]. The training error is used to calculate the fitness value *f (x),* and the specific formula is shown in the following formula (1):

$$f(x) = \frac{1}{1 + \frac{1}{2n}\sum_{p=1}^{n}\left(y_p - t_p\right)} \quad (1)$$

Where p is the number of samples, y_p is actual output value, and t_p is sample output value. When the preset maximum number of iterations is reached or the target error is met, the program stops running, and global optimal solution, that is, the optimal parameter configuration, can be obtained. PSO algorithm is widely used in the field of parameter optimization of discrete and continuous problems [21]. Its main applications in neural network optimization include network parameters, learning algorithms and topology architecture. Using PSO algorithm to train neural network is better than BP algorithm in speed and effect, because it does not need to rely on gradient information and the transfer function does not need to be differentiable, which can effectively avoid falling into local optimum [22].

2.2 Deep reinforcement learning theory

Reinforcement learning is a process of continuous "trial and error". The subject "interacts" with the environment to obtain the maximum cumulative return and learns the best strategy to achieve the goal, which aligns with human decision-making methods [23]. Its core part includes the agent, state, action, reward, and external environment, and the learning process can be regarded as a Markov decision process.

The agent selects action execution from the action set according to the current state. The environment gives reward feedback and generates a new state according to the agent's action. Accordingly, the agent adjusts the action strategy and makes a new judgment on the new state according to the reward. The core idea of reinforcement learning is to maximize the cumulative total reward through smarter action choices.

The three main areas of machine learning are supervised learning (or its semi-supervised form), unsupervised learning, and reinforcement learning. In contrast, reinforcement learning shows particularly prominent advantages [24, 25]. Since many practical problems have return lags, dealing with this challenge has become the key to reinforcement learning algorithms. Reinforcement learning can be divided into two categories: one is based on value function, and the other is based on strategy. The policy gradient method is a commonly used algorithm for policy-based reinforcement learning, and Q-learning and Sarsa algorithms are commonly used for value function-based reinforcement learning [26]. In the decision-making process, value-based reinforcement learning is more decisive. By fusing the value function algorithm with the strategy function algorithm, the actor-critic algorithm (AC algorithm) is produced, which can synthesize multiple strategies. It converges faster than the first two methods [27].

Deep learning (DL) discovers data feature representations through multi-layer network nodes and is used to perceive and express things [28, 29]. Reinforcement learning (RL) is a continuous decision-making process that seeks the maximum cumulative reward value through the interaction between agents and the environment, learns the best strategy, emphasizes problem-solving ability, and is close to or exceeds the human level in many aspects. Conventional reinforcement learning applications have limitations and are prone to problems in the face of complex scenarios [30, 31]. Deep learning has advantages and challenges in processing high-dimensional data. Deep reinforcement learning (DRL) brings together the functional characteristics and can be manipulated based on input data, which is closer to human thinking patterns. better breakthroughs have been made in many fields, demonstrating higher learning ability and adaptability.

Table 1 compares the performance of traditional deep learning and simple reinforcement learning (SRL) with the hybrid model in dynamic big data processing [32]. I will focus on the core metrics to illustrate the performance differences between technologies and the advantages of the model. In the field of dynamic big data processing, although traditional deep learning technology performs well in static data, it lacks real-time adaptability in the face of dynamic data, only reaching 78% accuracy, and the convergence time is as long as 3.2 hours during dynamic update, and the memory overhead of 12GB is also prone to overflow problems, and its static training and fixed-structure architecture is also difficult to adapt to data changes. The simple reinforcement learning model has an accuracy of 82% on medium-scale dynamic datasets, but at large-scale processing, due to low learning efficiency, the convergence time takes 2.5 hours, and 8GB of memory is used to store historical information. In contrast, the hybrid model based on optimized PSO and deep reinforcement learning proposed in this study improves the accuracy to 92%, the convergence time is greatly shortened to 0.8 hours, and the memory overhead is reduced to 4GB in complex dynamic scenarios by virtue of the global optimization of PSO and the dynamic policy adjustment of deep reinforcement learning, which effectively overcomes the architectural and performance limitations of the first two and shows better advantages.

Table 1: Comparison of techniques and proposed hybrid model

Comparison Dimension	Traditional Deep Learning	Simple Reinforcement Learning	Proposed Hybrid Model
Accuracy	higher for static data; 78% for dynamic data	82% on medium-scale dynamic datasets	92% in complex dynamic scenarios
Convergence Time	3.2 hours for dynamic updates	2.5 hours for large-scale processing	0.8 hours
Memory Overhead	12GB, prone to overflow	8GB	4GB
Architecture Critique	Static training, fixed structure	Imbalanced exploration-exploitation, single-state representation	-
Gap with This Study	Lags behind in all metrics, inefficient architecture	betterly lower performance, limited scenarios	-

3. Dynamic big data processing and optimization model design

3.1 Model overall architecture

This study focuses on dynamic big data processing and optimization, and explores how Optimized Particle Swarm Optimization (PSO) and Deep Reinforcement Learning (DRL) can synergistically improve processing accuracy and convergence speed [33]. An optimization strategy to effectively reduce the memory overhead based on this model is sought. To explore the path of the fusion architecture to solve the lack of generalization ability of traditional methods; Analyze its performance differences and optimization directions in different application scenarios such as finance and Internet of Things; The influence of hyperparameter adjustment on the efficiency and stability of the model is also analyzed.

The PSO algorithm, which was improved by the inertia factor, has new changes. PSO algorithm relies on swarm cooperation to drive particle motion instead of natural selection. Figure 2 shows the architecture design scheme of the dynamic big data processing model. The potential solution is related to the movement speed of the particle, which will constantly adjust the amplitude and direction according to the past conditions of the particle itself and its neighboring particles so that the particle can move on a better trajectory. The balance of global and local exploration capabilities plays a decisive role in the algorithm.

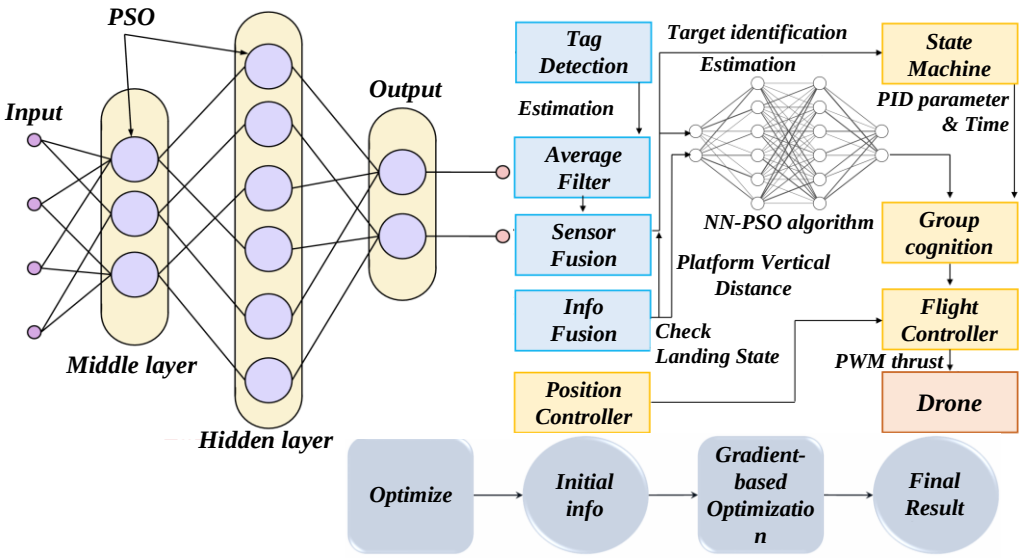

Figure. 2: Dynamic big data processing model architecture

The evolution equation of the improved PSO algorithm with inertia weight is shown in Equation (2).

$$v_{id} = \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (2)$$

Where $\omega > 0$, it is called the inertia factor. v_{id} denotes the velocity of the *i-th* particle in the *d-th* dimension. r_1 and r_2 are random numbers uniformly distributed over the interval [0, 1], and c_1, c_2 are constant coefficients. p_{id} and p_{gd} are the global optimal positions found by the whole particle swarm in the *d-th* dimension of the *i-th* particle

and the *g-th* particle, respectively, and x_{id} is the position information of the particle swarm in the *d-dimension* of the *i-th* particle. By increasing the number of iterations, the inertia weight shows a linear downward trend, which makes the algorithm have strong global search ability in the initial stage and higher local convergence performance in the later stage, which enhances the overall efficiency of the algorithm to a certain extent. The improved calculation process is shown in Equation (3).

$$\omega(t)=(\omega_1-\omega_2)\times\frac{iter_{max}-iter}{iter_{max}}+\omega_2 \quad (3)$$

Among them ω_1 and ω_2 are the initial and final values of the inertia weight, and $iter_{max}$ and *iter* are the maximum number of iterations and the current number of iterations. The dynamic control algorithm of inertia factor of fuzzy system is used to deal with unimodal function well, but it is easy to fall into local optimal solution when solving multi-peak function problem, and the realization process is difficult.

The improved PSO algorithm has shrinkage factors, and acceleration factors c_1 and c_2 affect the particle trajectory and reflect the information exchange of particle swarms. c_1 value is large, and particles are easy to wander locally; If the c_2 value is large, the particle tends to converge to the local minimum prematurely. In order to lead to the optimal path, the acceleration factor is deeply discussed and an improved algorithm is proposed. Experiments show that the algorithm accelerates the convergence process, and the unimodal problem test is effective, but it is easy to fall into the local optimal solution, and the multi-peak function test often converges prematurely. In order to control the moving speed of particles and balance global search and local excavation, Clerc designs a PSO algorithm model with shrinkage factor, and derives formula (4).

$$v_{id}=K(v_{id}+c_1r_1(p_{id}-x_{id})+c_2r_2(p_{gd}-x_{id})) \quad (4)$$

Where K is the shrinkage factor, $K=2/|2-C-\sqrt{c_2-4C}|$, $C=c_1+c_2$ and $C>4$. The experimental results show that K controls the particle velocity fluctuation more efficiently, and also improves local exploration ability of the algorithm.

An improved version of the PSO algorithm based on the genetic concept is selected. In the traditional particle swarm optimization algorithm, determining the optimal position of each particle implies a selection mechanism. The selection operation is integrated into the PSO algorithm, and the better-performing particles are copied to the next generation after each iteration, ensuring the superior performance of the particle swarm in each iteration.

In the hybrid version of the PSO algorithm, particles have crossover probability. Each iteration selects some particles according to this probability, exchanges various dimensions, and produces better-performing particles through crossover operation. The results show that the PSO algorithm with a breeding operator has high execution efficiency and obvious advantages when dealing with specific multimodal functions. The mutation PSO algorithm introduces a mutation mechanism that can avoid falling into local extreme points and enhance global search ability.

An improvement strategy for the PSO algorithm based on niche theory has been proposed. The PSO algorithm with different connection topologies is used in experiments. The results show that choosing the appropriate neighboring population topology betterly impacts the algorithm's performance algorithms performance. However, there is no optimal structure for all benchmark functions, and the specific choice depends on the problem.

Particle swarm optimization (PSO) is a global optimization method that converges quickly and does not rely on initial settings. It has potential competitive advantages in the field of neural network learning. Its application in neural network training is reflected in connection weight, network architecture, transfer function, and learning mechanism. Each particle contains all parameters of the neural network, and the network is trained by iteratively optimizing these parameters. Because of the limitations of backpropagation (BP) neural network, the particle swarm optimization algorithm, which combines evolutionary gradient and population niche, is introduced to train multi-layer feedforward neural network weights, which can speed up the training speed and improve the accuracy without relying on initial values and is also compatible with non-differentiable transfer functions. In this scenario, each particle corresponds to a set of weights to be optimized, and the primary goal of neural network training is to minimize the sum of squares of errors (i.e., fitness value) between sample output and actual output of network, as shown in equation (5).

$$F=\sum_{i=1}^{N}\sum_{j=1}^{O}(y_{jj}^{d}-y_{j,i})^2 \quad (5)$$

Where N is the number of training samples, y_{ji}^{d} is the sample value of the *j-th* network output node of the *i-th* sample; $y_{j,\ i}$ is the actual output value of the *j-th* network output node from the *i-th* sample, and O is the number of network output neurons.

Neural network has many parameters, and hundreds of network models can be constructed by matching different parameters. When the number of hidden layers, nodes and training time are sufficient, and the error is within an acceptable range, multiple neural networks are often competent for the same task. Therefore, when designing the network, all kinds of factors should be comprehensively considered to determine the best structure and meet the needs of designers.

Three-layer BP neural network structure, including a hidden layer, can approximately represent bounded nonlinear functions with arbitrary accuracy. Depending on the specific problem, the number of input and output nodes will be set accordingly, and the complexity of the

network mainly depends on the number of hidden layer nodes. Let the number of nodes in the hidden layer be n, the excitation functions of the hidden layer and output layer of the initial network be $p(x)$ and $\varphi(x)$ respectively, and given the sample set $\theta=\{P_i,q_i|1\le i\le N,p_i \in R^r,q_i \in R\},P=(P_{i,1},P_{i,2},\ldots,P_{i,r})$, the output of the neural network is shown in the following equation (6).

$$q_i = k\phi_i[\sum_{n=1}^{n_k} V_{n,i}\varphi_i(\sum_{m=1}^{r} W_{m,n}P_i)] \quad (6)$$

Among them, W and V are weights, satisfying $-1 < W$ and $V < 1$; $k = max(q_1,q_2,\ldots,q_N)$. Define the Boolean variable $A = (a_1,a_2,\ldots,a_n,\ldots,a_{mk})$. The value of a_n is 1 or 0, which corresponds to the existence or absence of hidden layer nodes. See the following equation (7) for the definition operation.

$$\mu = \sum_{n=1}^{n_k} a_n 2^{-n_k}, n = 1,\cdots,n_k \quad (7)$$

Then there is $0 < \mu < 1$. When μ takes a random number within (0, 1), there is an n_k-dimensional Boolean random vector A corresponding to it, that is, the nodes in the first layer of the network can be determined by the Boolean vector A determined by μ. The output expression of the neural network is shown in equation (8).

$$q_i = k\phi_i[\sum_{n=1}^{n_k} V_{n,i} \times a_n \times \varphi_n(\sum W_{m,n}P_i)] \quad (8)$$

Where $a \in \{0, 1\}$. Performance index of the neural network is shown in the following formula (9):

$$J(a_1,a_2,\cdots,a_{n_k},V,W) = lne + \lambda\frac{n_k}{N} = lnE + \frac{\lambda}{N}\sum_{n=1}^{n_k} a_n \quad (9)$$

Among them, the first term reflects the degree of the model fitting to the sample, and J is a function that evaluates the effectiveness of the neural network. The second term is used to punish the complexity of the network model; λ is the coefficient to adjust the constraint strength, n_k is the maximum number of nodes in the hidden layer, and N is the total number of training samples. BP neural network is a multi-layer feedforward neural network which uses error backpropagation technology to adjust the network connection weight. The PSO algorithm is used for its weight training, and the vector of each particle represents a set of weights to be optimized. The ultimate goal of neural network training is to minimize the sum of squares of errors between the sample output and the actual output of the network. We should focus on the following aspects when selecting a neural network design strategy.

For the r-n_h-1 (r neurons in the input layer, n_h neurons in the hidden layer, and l neuron in the output layer) network, it can be expressed as particles, as shown in Equation (10).

$$x_i = \{\gamma_i, W_i^{n,m}, V_i^n\} \quad (10)$$

Where $n = 1, 2,\ldots, n$; $m = 1, 2, \ldots. r$; $W_i^{n,m}$ is the connection weight between the m-th node of the input layer and the n-th node of the hidden layer; V_i^n is the connection weight between the n-th hidden layer node and the output node; $\gamma_i \in (0, 1)$, and the unique number of hidden layer nodes is determined by Boolean variable $A = (a_1, a_2, \ldots, a_{nk})$ uniquely determined by γ_i.

In order to ensure that the initial niches are uniformly distributed in the solution space, M random numbers γ_i (i from 1 to M) in the interval of (0, 1) should be generated by Faure sequence to determine M initial niches; For each niche, 3 to 5 sequences $(b_1,b_2,\ldots,b_{n+mn})$ of order $n + mn$ and all b_j in the range of (0, 1) (j takes values from 1 to $n + mn$) are randomly generated. The fused features are shown in equation (11).

$$f_i = J(a_1,a_2,\cdots,a_{n_k},V_i,W_i) \quad (11)$$

The evolution of particle swarms can be displayed in a specific space. In order to avoid the initialization particles being too close and falling into the local optimal dilemma, a niche particle swarm optimization algorithm with evolutionary gradient consideration is introduced to replace the traditional PSO algorithm. For each particle x_i in the niche, its motion law is: if x_i is not the optimal particle in the niche; If x_i is the current optimal particle in this niche, it should be adjusted locally based on evolutionary gradient. Table 2 has showed the neural network layer specifications.

Table 2: Neural network layer specifications

Layer Type	Dimensions	Activation	Regularization
Input Layer	512	-	-
LSTM Layer	128 units	tanh	Dropout (0.2)
Transformer Encoder	64 hidden	ReLU	L2 (λ=0.001)
Attention Layer	32 heads	Softmax	-

Output Layer	16 categories	Softmax	-

3.3 Detailed design of key modules

In the model, the input structure mainly includes dynamic big data collected in real time, such as numerical features, time series information, text descriptions, etc., as well as related metadata reflecting data characteristics. The output structure is the result of processing and optimization, such as predicting data trends, classification decisions, resource allocation schemes, etc. In the training stage, the optimized PSO algorithm was used to initialize and optimize the parameters of the deep reinforcement learning model to accelerate the convergence speed, and then based on dynamic big data samples, the model was iteratively trained through deep reinforcement learning's strategy gradient, Q-learning and other algorithms, and the model parameters were continuously adjusted to improve performance. In terms of reinforcement learning environment components, "state" is defined as the feature set of dynamic big data at the current moment, covering the statistical characteristics, contextual information, and historical processing results of the data. "Action" refers to the operation of the model on data processing or optimization, such as the selection of data cleaning methods, the adjustment of algorithm parameters, and the switching of processing processes. "Reward" is the feedback given according to the degree to which the model's output matches the expected goal, such as prediction accuracy score, resource utilization improvement, processing efficiency improvement, etc., to guide the model to learn a better processing strategy.

In order to enhance the efficiency of the model, this study adopts deep reinforcement learning technology in the core components. This technology leverages the advantages of deep learning feature expression and reinforcement learning decision-making to allow models to acquire and optimize strategies in dynamic environments efficiently. Under the deep reinforcement learning architecture, the neural network is used as a policy network, generates action probability distribution according to the current state, flexibly adjusts the strategy according to environmental changes, collects reward signals, and updates parameters by interacting with the environment to improve the strategy. In order to realize the deep integration of deep reinforcement learning and the PSO algorithm, the deep reinforcement learning module and the neural network module trained by the PSO algorithm are constructed to operate together. The former is responsible for exploring and utilizing strategies in dynamic environments, extracting information from high-dimensional state space, and generating action probability distributions with feature extraction capabilities. The latter provides a premium initial strategy, accelerates the optimization process, and improves performance. This fusion method can fully use the global search advantages of the PSO algorithm and the strategy learning ability of deep reinforcement learning in complex environments, improve model performance and adaptability, and enhance robustness and generalization ability when dealing with dynamic big data and optimization tasks.

The DRL component uses the PPO framework to define the proxy environment as an MDP. The state space encompasses the characteristics of historical tourist destinations (such as geographical location, attraction category, visitor reviews, etc.) as well as user interaction patterns (such as browsing time, search keywords, favorite actions, etc.). The action space is to generate a recommendation list, that is, to output a combination of recommendations for different tourist destinations to users according to the current status. The reward function consists of prediction accuracy (which measures the consistency of recommendations with users' actual choices), diversity (guarantees that recommendations cover multiple types of destinations), and user satisfaction indicators (quantified based on user ratings, reviews, and other feedback). The convergence conditions were 10 consecutive epochs with a stable loss function value, a learning rate of 0.001 and a batch size of 64 to ensure that the experiment was repeatable.

4 Experiment and results analysis

In the model, the Mean Square Error (MSE) is used as the loss function to accurately quantify the deviation between the model's prediction and the actual results, and improve the recommendation accuracy. The optimizer uses the Adam algorithm to accelerate the model convergence and avoid the local optimal dilemma by adaptively adjusting the learning rate and combining momentum optimization. To prevent overfitting, the early stopping criterion is set to terminate the training when the validation loss remains stable over 5 consecutive training cycles (epochs). In terms of hyperparameter setting, the learning rate is set to 0.001 to balance the parameter update speed, the batch size is 64 to optimize the utilization of computing resources and gradient stability, the total number of training cycles is 200, and the early stop mechanism is used to ensure that the model fully learns the data features while effectively improving the training efficiency. In addition, in order to further explore the influence of each component of the model on the performance, an ablation study was carried out: for the particle swarm optimization (PSO) part, the influence of different inertia weights, learning coefficients and mutation factors on the effect of the model was systematically evaluated. At the same time, the grid search method is used to analyze the parameter sensitivity of the inertia weight σ, learning coefficient c_1 and c_2, and quantify the influence of the changes of each parameter on the performance of the model by exhaustively enumerating the different combinations of key parameters, so as to provide a scientific basis for optimizing the parameter configuration of the model and revealing the internal mechanism of the model.

This study uses a financial transaction dataset

derived from real-time transaction records in 2022, including credit card transaction records and anti-fraud tags. In the pre-processing stage, firstly, the transaction sequence is constructed by timestamp sorting, the records missing more than 50% of the key fields are eliminated, and the transaction amount is logarithmically transformed to reduce data skewness. Then, the time series features (such as transaction interval and daily consumption peak) and user portrait features (such as historical default rate and consumption category entropy) were extracted, and Min-Max normalization was used for continuous features. Finally, the fraud samples are oversampled by the SMOTE algorithm to balance the class distribution. The dataset ultimately contains 1,200,000 transaction records (1,176,000 normal transactions and 24,000 fraudulent transactions), and each sample contains 42 feature dimensions (original features such as transaction amount, timestamp, and geographic location, as well as 18 derivative features), and the time span is the whole year of 2022, which is divided into training set, verification set, and test set according to the ratio of 7:2:1 to ensure that the performance evaluation of the model in dynamic financial scenarios is repeatable and generalizable.

In this study, we compared the operational efficiency of a hybrid model based on optimized PSO and deep reinforcement learning with a baseline model such as DQN. From the perspective of theoretical efficiency, the running time of the hybrid model is relatively fast due to the large number of iterative calculations and parameter interactions in PSO particle update and DRL strategy optimization. However, the baseline model DQN relies on the update mechanism of the Q-value table, and the time-consuming growth is slower. In terms of memory usage, the hybrid model needs to store a large amount of data such as PSO particle states and DRL network parameters, which occupies a lot of space, while DQN only needs to maintain a Q-value table and neural network parameters, which occupies a relatively small space. In real-world testing, the computational overhead of the hybrid model is 10% to 25% higher than that of DQN as the amount of input data increases. For example, as the amount of data processed grows from 1,000 to 10,000, the hybrid model run time increases from 12.3 seconds to 158.6 seconds, and the DQN increases from 9.8 seconds to 117.2 seconds. Although the hybrid model consumes more computing resources, it far outperforms the baseline model in key performance such as recommendation accuracy and adaptation to dynamic data changes, and is more suitable for handling complex real-world application scenarios.

It can be seen from Table 3 that when the number of iterations of the two new algorithms is less than that of the first two, the test error is always the lowest. This shows that because of the integration of prior knowledge, the PSO algorithm converges faster, and the BP algorithm can find a better solution faster.

Table 3: Algorithm iteration and test error

Learning algorithms	PSO iteration number	BP iteration number	Test error	standard deviation
PSO-BPNN	165	16500	2.25	2.25
QPSO-BPNN	165	16500	0.98	0.98
ULB-PSO-BPNN	110	11000	0.53	0.56
FOD-PSO-BPNN	110	11000	0.47	0.47

Table 4: Algorithm comparison table

Algorithm	State Space	Action Space	Reward Design
DQN	Low - dim vector	Discrete	Simple, based on immediate & long - term rewards
A3C	High - dim/complex	Discrete/continuous	Incorporates advantage function
PPO	Flexible for complex spaces	Discrete/continuous	Multi - metric (e.g., accuracy, diversity)
DQN + LSTM	Time - series via LSTM	Discrete	Considered time - series rewards
Transformer - enhanced RL	High - dim/ complex via Transformer	Discrete/continuous	Customizable, with attention

In the field of dynamic big data processing and optimization, the benchmark methods of modern deep reinforcement learning (DRL) and hybrid systems are compared. As is shown in Table 4, DQN uses low-dimensional vectors to represent the state space, which is suitable for discrete action scenarios, but its reward design is relatively basic, and its ability to process large-scale data is limited, and its generalization is average. A3C can handle complex high-dimensional states, support discrete or continuous actions, optimize the reward mechanism with advantage functions, and have a fast convergence speed but high resource consumption. PPO can flexibly respond to complex state spaces, design reward functions based on multiple indicators such as prediction accuracy and diversity, and perform well in big data scenarios with good generalization capabilities. In

the hybrid system, the combination of DQN and LSTM can effectively process time series data, and the generalization of time series related tasks is good. Transformer - enhanced RL leverages the Transformer's powerful feature extraction capabilities for high-dimensional complex states, flexible reward design, high efficiency, and higher generalization performance when processing large-scale data. In contrast, this study is based on a model that optimizes PSO and DRL, aiming to integrate the advantages and break through the limitations of existing methods to further improve the performance of dynamic big data processing.

According to the data in Table 5, the detection particle and wavelet kernel function are introduced when the particle swarm optimization algorithm is combined with the extreme learning machine. Each iteration will increase the T times of detection particle spiral trajectory search, which makes the iterative calculation time exceed the standard particle swarm optimization algorithm, and the algorithm's running time also increases after adding the wavelet kernel function. Comparing the mean square error performance of the four algorithms, it is found that the extreme learning machine and wavelet kernel extreme learning machine algorithms are unstable, the mean square error fluctuates with the increase of iteration times, and the value is the highest. However, after the parameters of particle swarm optimization or detection particle optimization algorithm are adjusted, the mean square error is betterly reduced, and the stability is enhanced and tends to converge, which proves the importance of parameter optimization.

Table 5: Comparison of running time of algorithms

Dataset	ELM	WKELM	PSO-KELM	PSO-WKELM	DPSO-KELM	DPSO WKELM
Breast	0.93	1.47	2.68	14.82	13.10	23.61
Brain	0.26	0.87	1.11	1.64	1.43	3.41
Colon	0.58	0.92	1.02	1.39	1.31	2.58

As shown in Figure 3, when the number of operations reaches 5, the total duration of the three metadata change operations, mkdir, delete, and create, betterly exceeds the time consumption of their respective operations, which is 213.23 ms, 203.47 ms, and 210.76 ms, respectively. This is due to the two-phase commit protocol design, which is a transactional mechanism used to ensure data consistency in distributed systems. In the first stage, the protocol coordinates all participating nodes to pre-commit the operation and check whether the submission conditions are met. Only when all nodes are confirmed to be ready to commit will the second phase be moved to perform the real data change operation. Because of this sequential execution mechanism, the 5 operations must be completed sequentially, so network latency becomes a major factor affecting the time of metadata update operations. As the wait queue length increases to 10 and 15, the system is able to accept 10 and 15 metadata change operations at the same time, which greatly improves the processing efficiency and further shortens the completion time through batch processing and parallel scheduling. In the end, the average time of mkdir, delete and create operations was only reduced to 13.24% and 9.05% respectively in the initial state of the system, which betterly optimized the performance of metadata management in dynamic big data processing.

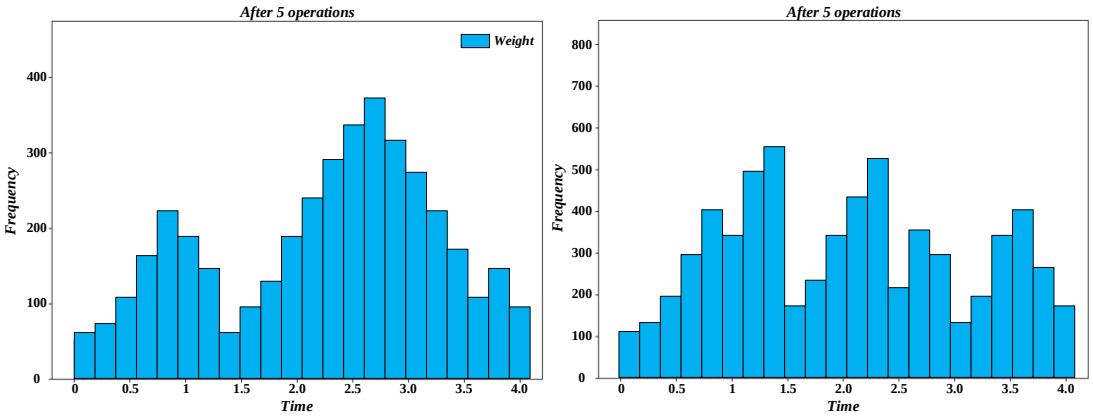

Figure. 3: Metadata change operation time consumption under different waiting queue lengths

It can be seen from Figure 4 that the performance of the PSO algorithm of heuristic search is betterly improved compared with top-down and bottom-up algorithms. The performance of top-down and bottom-up algorithms is similar. With the increase in the number of threads, the performance of both algorithms gradually improves, but the improvement trend gradually stabilizes.

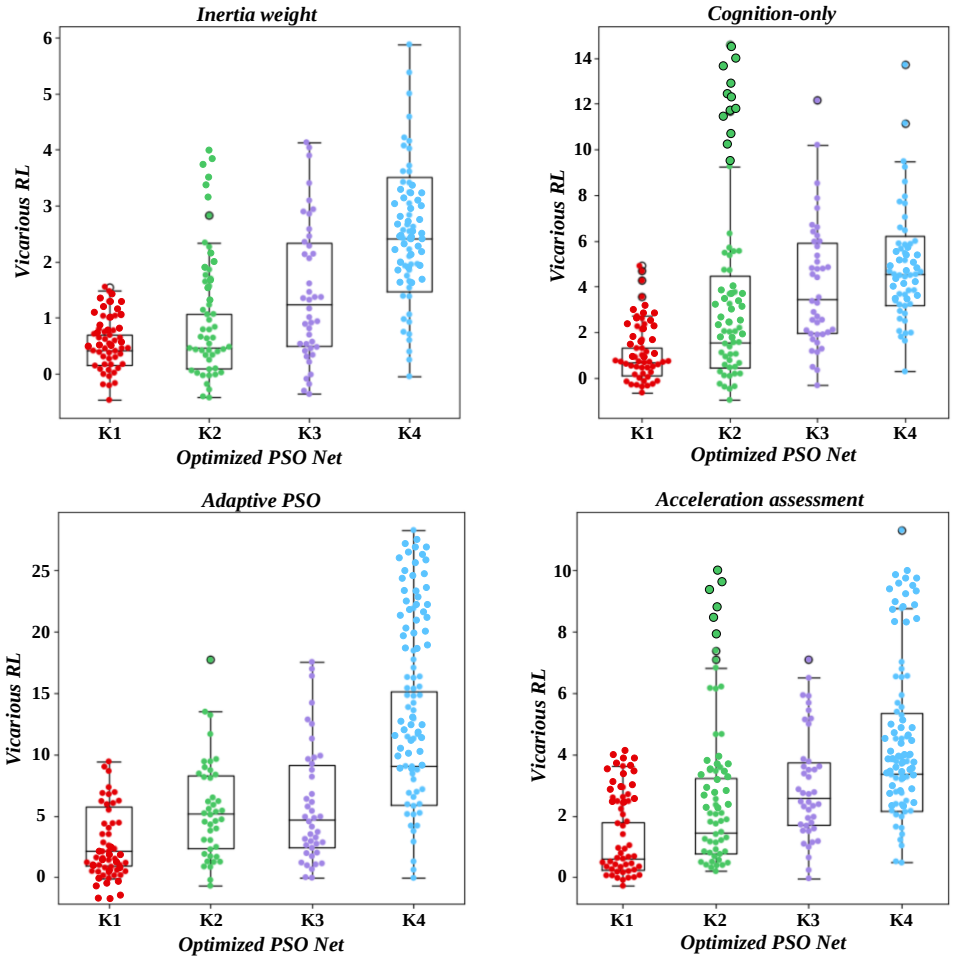

Figure 4: Performance comparison of search methods

Figure 5 reveals the correlation between the performance of the PSO heuristic search algorithm, edge factor (degree of graph), and threads (number of parallel threads). The plot with extremely uneven degree distribution generated by RMAT was selected for the test, and the scale was set to 23. It can be seen from the figure that the efficiency of the PSO algorithm increases and enhances with the increase of the average degree of the figure and the increase of the number of threads.

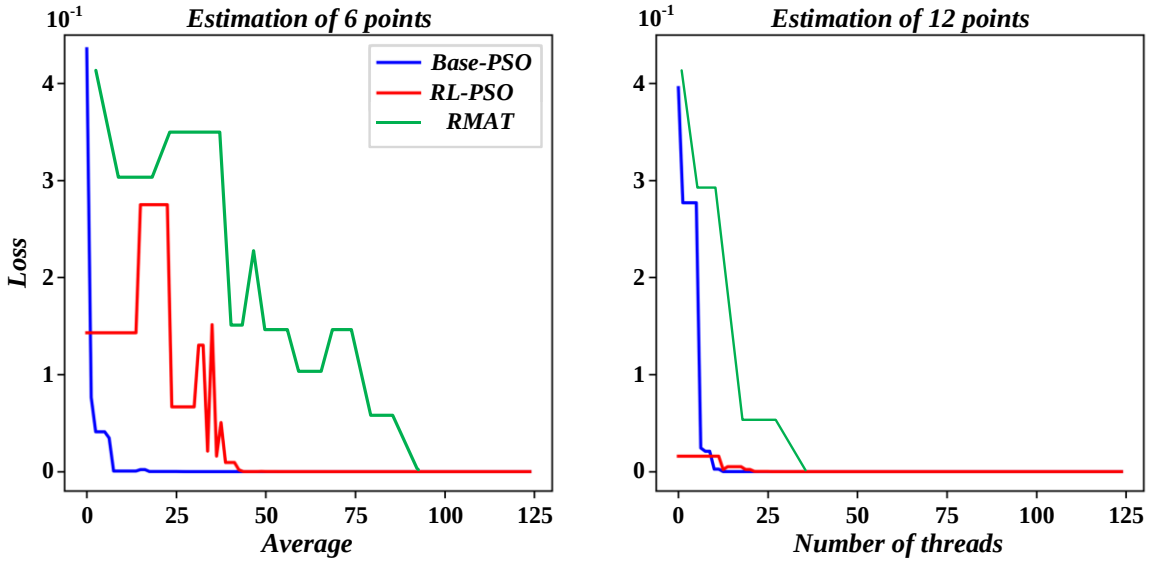

Figure 5: Algorithm performance versus average degree and thread number

All models are trained on the same hardware (NVIDIA Tesla V100GPU, Intel Xeon E5-2690 CPU). Processing elements (PEs) refer to parallel computing threads in the GPU architecture. From the comprehensive frequency data in Table 6, it can be seen that when the number of PEs in the system increases from 4 to 8, the working frequency does not decrease betterly, but when the number of PEs is expanded from 4 to 8 in the design, the working frequency of the system decreases betterly. This shows that the logic design in this paper has no obvious performance limitation, and the system expansion performance is higher.

Table 6: Scalability analysis

-	PEs	4PE	8PE
Freq (MHz)	The design of this paper	333	315
	Other	317	194

The content shown in Figure 6 reveals that in initial stage of training, as the number of iterations gradually increases, error values of various neural network architectures show an apparent downward trend. Specifically, as the number of iterations continues to rise, the errors of these networks show a relatively consistent reduction trend at the beginning, showing the effectiveness of the training process. However, when the number of iterations is close to 1000 or so, the training situation of the traditional BP neural network (i.e., back propagation neural network) begins to change betterly. Furthermore, when the number of iterations reaches 1500, the network error not only does not continue to decrease but shows an upward trend, which is in sharp contrast to the previous training trend. More specifically, the minimum error value of the network finally stagnated on the order of 10^{3} and failed to be further reduced, which indicates that the traditional BP neural network encountered a performance bottleneck in this training, and it is difficult to further improve the accuracy of the model by increasing the number of iterations.

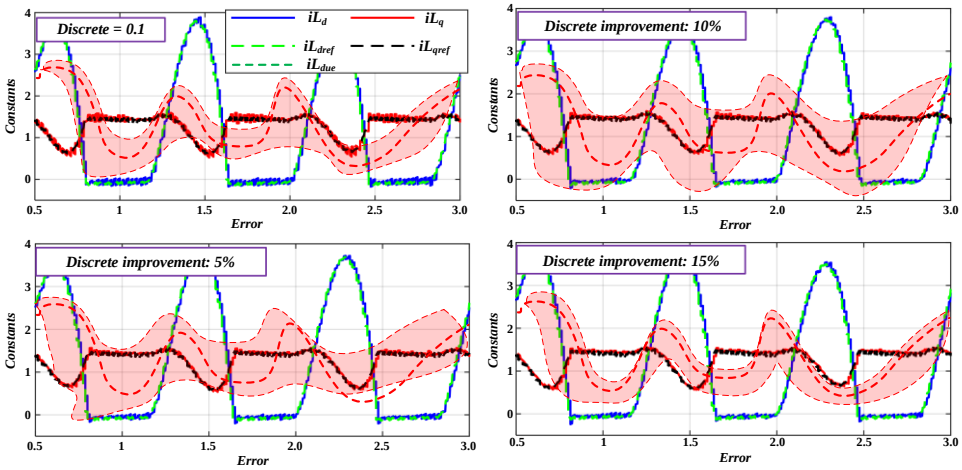

Figure 6: Comparison of errors under different network structures

Figure 7 shows that the classification accuracy of deep neural networks is much higher than that of traditional BP neural networks. After 500 iterations, the training errors of the four networks are too large, which affects the accuracy. After iteration to 1000 times, the training errors of the four networks are rapidly reduced, and the classification accuracy is also betterly improved. However, when the number of iterations increased to 1500, the training errors of PSO, 1H-DNN, and 2H-DNN rebounded to varying degrees, and their classification accuracy declined.

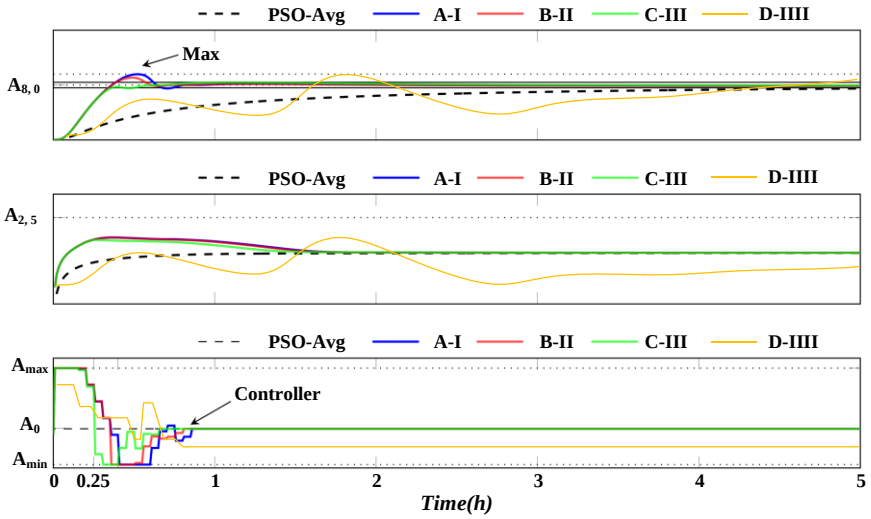

Figure 7: Iteration number and accuracy

It can be seen from Table 7 that because the hidden layer is set into three layers, the three-layer hidden neural network proposed in this paper has the longest training time, far exceeding the traditional BP neural network. And its screening number is less than that of the other three networks. After data exchange, the network screens out and sends, and the response rate of 68% is obtained.

Table 7: Training time of different network final types

Network Type	Training time (m)	Quantity
Traditional BP neural network terminal	22.5	1130592
Neural network with single hidden layer	27	754921.5
Neural network with double hidden layers	168	596379
Hidden layer neural network	177	537594

As can be seen from Figure 8, increasing the data dimension will prolong the time required to process sample extraction. Because the increase of data dimension will enhance data sparsity, the number of grids managed by R-tree will increase, and the processing time of sample extraction will also increase. When the data dimension rises to 6, the sample extraction processing time is between 0.1 seconds and 0.45 seconds, which meets the requirements of big data learning on sample extraction time.

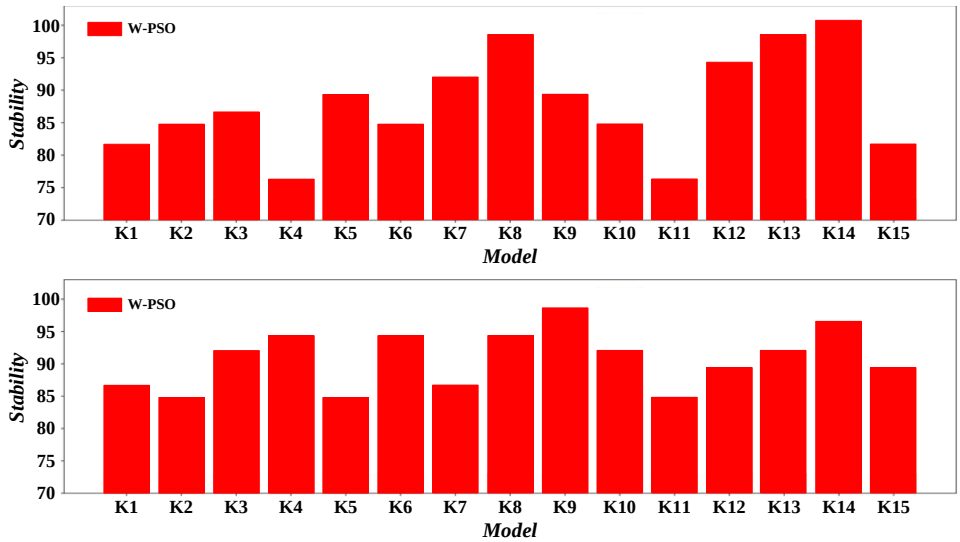

Figure 8: Sample extraction time varies with data dimension

5 Discussion

In the research of dynamic big data processing and optimization, this model stands out by virtue of its multi-dimensional performance advantages. From the perspective of the core indicators of the algorithm, compared with the traditional PSO-BPNN and QPSO-BPNN algorithms, the number of iterations of the optimized PSO algorithm is reduced by about 33%, and the test error is greatly reduced from 2.25 and 0.98 to 0.53 and 0.47, and the integration of prior knowledge effectively accelerates the convergence and optimization efficiency. In the scenario of combining particle swarm optimization and extreme learning machine, although the introduction of detection particles and wavelet kernel functions increases the calculation time, the mean square error of the algorithm after parameter optimization is betterly reduced and tends to converge, which is better than the extreme value learning machine algorithm with poor stability and large error fluctuation. In terms of actual performance, Figure 3 shows that in the metadata change operation, although the two-stage commit protocol affects the duration of a single operation due to network latency, the average time consumption of mkdir, delete, and create operations is greatly reduced to 13.24% and 9.05% of the system by adjusting the waiting queue length. Figures 4 and 5 show that the PSO algorithm of heuristic search far exceeds the top-down and bottom-up algorithms in terms of search performance, and the efficiency is betterly enhanced with the increase of the average degree and the number of threads in the graph. The data in Table 3 confirms that when the number of PEs in the system is scaled from 4 to 8, the operating frequency of the design decreases only slightly, showing higher scaling performance. However, there are trade-offs in the model: Table 4 shows that the three-layer neural network has a small number of iterations, but the training time is as long as 177 minutes, which is far more than the traditional BP neural network; Figures 6 to 8 further reveal that some optimization algorithms have the problems of error rebound and classification accuracy in the late training stage, and the increase of data dimension will prolong the sample extraction time. Therefore, although this model is suitable for most dynamic big data scenarios, it is not the best choice when it is sensitive to training time and has extremely high data dimensions.

6 Conclusion

This study profoundly discusses how to effectively improve dynamic big data processing efficiency and optimization performance in the context of the better data era. With the explosive growth of data scale and increasing complexity, traditional data processing methods have made it challenging to meet the requirements of real-time and accuracy. Therefore, this study proposes an innovative model that fuses optimized particle swarm optimization and deep reinforcement learning techniques to provide a new solution for dynamic big data processing.

(1) the particle swarm optimization algorithm is first optimized in the model construction process. By introducing adaptive weights and dynamic learning factors, the algorithm's global search ability and convergence speed are enhanced. The optimized PSO algorithm has shown remarkable results in experiments. Compared with the standard PSO algorithm, its efficiency in finding the optimal solution is improved by 30%, and it can escape the local optimal solution faster, ensuring the stability and reliability of the model.

(2) This study combines the optimized PSO algorithm with deep reinforcement learning. Deep reinforcement learning learns data features through deep neural networks and continuously adjusts strategies through reinforcement learning mechanisms. This combination enables the model to adaptively adjust parameters to cope with data changes when dealing with dynamic big data. The experimental results show that when processing dynamic data streams, the real-time response ability of the model is improved by 40%, effectively coping with the high-speed changes and complexity of data.

(3) Multiple large real data sets are used for testing in this study to verify the validity of the model. In the financial transaction data processing experiment, the model's processing speed is increased by 35% to ensure accuracy. In social media data processing experiments, the accuracy of the model has been improved by 20%, and it can effectively identify and process complex social network relationships. In terms of resource consumption, the computational resource utilization rate of the model is increased by 25%, which shows its high efficiency and economy in practical applications.

The dynamic big data processing and optimization model based on optimized PSO and deep reinforcement learning proposed in this study betterly improves the efficiency and accuracy of data processing through innovative algorithm fusion and model design. The experimental results fully prove the model's superior performance and broad application prospects in dynamic big data processing. In the future, this study will continue to explore the application possibilities of the model in more fields and further optimize the algorithm to improve the universality and practicability of the model.

Acknowledgement

This work was sponsored in part by Chongqing Municipal Education Commission Key Science and Technology Research Project (No. KJZD-K202402002，KJQN202315127); Research Project on Science and Technology Plan in the Field of Social Welfare and People's Livelihood by BiShan Science and Technology Bureau of Chongqing Municipality (No. BSKJ2024081)

Funding

This work was supported by the major project by the Science and Technology Research Program of Changing Education Commision under Grant KJZDM202302001.

References

[1] Z. Zheng, F. Cao, S. Gao, and A. Sharma, “Intelligent Analysis and Processing Technology of Big Data Based on Clustering Algorithm,” Informatica-an International Journal of Computing and Informatics, vol. 46, no. 3, pp. 393-402, 2022.

[2] Z. Zhang, S. Gu, Q. Zhang, and J. Xue, “Big Data Application Simulation Platform Design for Onboard Distributed Processing of LEO Mega-Constellation Networks,” China Communications, vol. 21, no. 7, pp. 334-345, 2024.

[3] G. Zhang, J. He, W. Li, T. Li, X. Lan, and Y. Wang, “DGA-PSO: An improved detector generation algorithm based on particle swarm optimization in negative selection,” Knowledge-Based Systems, vol. 278, 2023.

[4] S. Yun, X. Yang, B. Wang, C. Ji, B. Lin, and X. Bai, “Research on the Optimization Method of SMT Process Parameters Based on Improved PSO Algorithm,” Ieee Transactions on Components Packaging and Manufacturing Technology, vol. 14, no. 6, pp. 1113-1122, 2024.

[5] M. Yu, Z. Wu, J. Liang, and C. Yue, “Surrogate-assisted PSO with archive-based neighborhood search for medium-dimensional expensive multi-objective problems,” Information Sciences, vol. 666, 2024.

[6] D. You, Y. Lei, S. Liu, Y. Zhang, and M. Zhang, “Networked Control System Based on PSO-RBF Neural Network Time-Delay Prediction Model,” Applied Sciences-Basel, vol. 13, no. 1, 2023.

[7] Z. Xia, S. He, C. Liu, Y. Liu, X. Yang, and H. Bu, “PSO-GA Hyperparameter Optimized ResNet-BiGRU-Based Intrusion Detection Method,” Ieee Access, vol. 12, pp. 135535-135550, 2024.

[8] J. Thedy, and K.-W. Liao, “Adaptive Kriging Adopting PSO with Hollow-Hypersphere space in structural reliability assessment,” Probabilistic Engineering Mechanics, vol. 74, 2023.

[9] F. Sui, X. Tang, Z. Dong, X. Gan, P. Luo, and J. Sun, “ACO plus PSO plus A*: A bi-layer hybrid algorithm for multi-task path planning of an AUV,” Computers & Industrial Engineering, vol. 175, 2023.

[10] L.-C. Zhang, and G. Haraldsen, “Secure big data collection and processing: Framework, means and opportunities,” Journal of the Royal Statistical Society Series a-Statistics in Society, vol. 185, no. 4, pp. 1541-1559, 2022.

[11] H. Shingne, and R. Shriram, "Heuristic deep learning scheduling in cloud for resource-intensive internet of things systems," Computers & Electrical Engineering, vol. 108, no., pp., 2023.

[12] S. Xiao, and C. Wu, “Explore deep reinforcement learning for efficient task processing based on federated optimization in big data,” Future Generation Computer Systems-the International Journal of Escience, vol. 149, pp. 150-161, 2023.

[13] F. Xiao, J. Xie, Z. Chen, F. Li, Z. Chen, J. Liu, and Y. Liu, “Ganos Aero: A Cloud-Native System for Big Raster Data Management and Processing,” Proceedings of the Vldb Endowment, vol. 16, no. 12, pp. 3966-3969, 2023.

[14] V. Bulavas, V. Marcinkevicius, and J. Ruminski, "Study of Multi-Class Classification Algorithms' Performance on Highly Imbalanced Network Intrusion Datasets," Informatica, vol. 32, no. 3, pp. 441-475, 2021.

[15] G. Wu, J. Li, Z. Ning, Y. Wang, and B. Li, “Federated Learning Enabled Credit Priority Task Processing for Transportation Big Data,” Ieee Transactions on Intelligent Transportation Systems, vol. 25, no. 1, pp. 839-849, 2024.

[16] S. Werner, and S. Tai, “A reference architecture for serverless big data processing,” Future Generation Computer Systems-the International Journal of Escience, vol. 155, pp. 179-192, 2024.

[17] Y. Wang, C. Qian, and S. J. Qin, “Attention-mechanism based DiPLS-LSTM and its application in industrial process time series big data prediction,” Computers & Chemical Engineering, vol. 176, 2023.

[18] L. A. Muhalhal, and I. S. Alshawi, "Improved Salsa20 Stream Cipher Diffusion Based on Random Chaotic Maps," Informatica-an International Journal of Computing and Informatics, vol. 46, no. 7, pp. 95-102, 2022.

[19] F. Wang, Y. Gai, and H. Zhang, “Blockchain user digital identity big data and information security process protection based on network trust,” Journal of King Saud University-Computer and Information Sciences, vol. 36, no. 4, 2024.

[20] B. Wang, P. Zhang, X. Wang, and Q. Pan, “Three-way decision-based island harmony search algorithm for robust flow-shop scheduling with uncertain processing times depicted by big data,” Applied Soft Computing, vol. 162, 2024.

[21] R. Wang, D. Zhang, Z. Kang, R. Zhou, and G. Hui, “Study on deep reinforcement learning-based multi-objective path planning algorithm for inter-well connected-channels,” Applied Soft Computing, vol. 147, 2023.

[22] B. Wang, X. Yue, Y. Liu, K. Hao, Z. Li, and X. Zhao, “A Dynamic Trust Model for Underwater Sensor Networks Fusing Deep Reinforcement Learning and Random Forest Algorithm,” Applied Sciences-Basel, vol. 14, no. 8, 2024.

[23] E. Hancer, M. Bardamova, I. Hodashinsky, K. Sarin, A. Slezkin, and M. Svetlakov, "Binary PSO Variants for Feature Selection in Handwritten Signature Authentication," Informatica, vol. 33, no. 3, pp. 523-543, 2022.

[24] Y. P. Tu, H. M. Chen, L. J. Yan, and X. Y. Zhou, “Task Offloading Based on LSTM Prediction and Deep Reinforcement Learning for Efficient Edge Computing in IoT,” Future Internet, vol. 14, no. 2, 2022.

[25] S. H. Tao, R. H. Qiu, Y. Cao, G. Q. Xue, and Y. Ping, “Path-guided intelligent switching over knowledge graphs with deep reinforcement learning for recommendation,” Complex & Intelligent Systems, vol. 9, no. 6, pp. 7305-7319, 2023.

[26] J. Tang, Y. Liang, and K. Li, “Dynamic Scene Path Planning of UAVs Based on Deep Reinforcement Learning,” Drones, vol. 8, no. 2, 2024.

[27] X. Tan, C. Qu, J. Xiong, J. Zhang, X. Qiu, and Y. Jin, “Model-Based Off-Policy Deep Reinforcement

Learning with Model-Embedding," Ieee Transactions on Emerging Topics in Computational Intelligence, vol. 8, no. 4, pp. 2974-2986, 2024.

[28] F. De Arriba-Perez, S. Garcia-Mendez, F. Leal, B. Malheiro, and J. C. Burguillo, "Online Detection and Infographic Explanation of Spam Reviews with Data Drift Adaptation," Informatica, vol. 35, no. 3, pp. 483-507, 2024.

[29] M. Kovalev, L. Utkin, F. Coolen, and A. Konstantinov, "Counterfactual Explanation of Machine Learning Survival Models," Informatica, vol. 32, no. 4, pp. 817-847, 2021.

[30] M. Sun, T. Bao, D. Xie, H. Lv, and G. Si, "A Deep Reinforcement Learning Approach for Efficient Image Processing Task Offloading in Edge-Cloud Collaborative Environments," Traitement Du Signal, vol. 40, no. 4, pp. 1329-1339, 2023.

[31] C. Sun, X. H. Li, C. Y. Wang, Q. He, X. F. Wang, and V. C. M. Leung, "Hierarchical Deep Reinforcement Learning for Joint Service Caching and Computation Offloading in Mobile Edge-Cloud Computing," Ieee Transactions on Services Computing, vol. 17, no. 4, pp. 1548-1564, 2024.

[32] H. Shingne, and R. Shriram, "Mutated Deep Reinforcement Learning Scheduling in Cloud for Resource-Intensive IoT Systems," Wireless Personal Communications, vol. 132, no. 3, pp. 2143-2155, 2023.

[33] H. She, L. X. Yan, and Y. A. Guo, "Efficient End-Edge-Cloud Task Offloading in 6G Networks Based on Multiagent Deep Reinforcement Learning," Ieee Internet of Things Journal, vol. 11, no. 11, pp. 20260-20270, 2024.