International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18 (2026) pp. 1-15
© Cerebration Science Publishing

https://doi.org/10.70917/ijcisim-2026-0134
Article

Research on User Preference Modeling and Data Enhancement Based on Generative Adversarial Networks in Precision Recommendation for E-Commerce Platforms

Lanyan Yang *

Computer and Information Engineering College, Guizhou University of Commerce, Guiyang, Guizhou, 550014, China; ylanyang12@sina.com

Abstract: With the advent of the intelligent internet era, e-commerce has also experienced explosive growth, leading to intensified competition among various e-commerce platforms. To accurately identify user preferences on e-commerce platforms and eliminate irrelevant items, this paper proposes a user-demand-based generative adversarial recommendation algorithm based on generative adversarial networks (GANs). This algorithm consists of multiple generators, with a discriminative model evaluating the generated items and providing feedback to the generative model to continuously improve it, ultimately recommending items with high similarity to the generated items. Comparative experiments were conducted on two datasets, and the model training was completed using the proposed model. Experimental simulations were conducted on the MovieLens-100K dataset. The experimental results indicate that the proposed algorithm can effectively improve accuracy. UR-GAN has advantages such as fast convergence speed and a stable training process, making the practical value of the recommendation system based on UR-GAN relatively high.

Keywords: generative adversarial network; recommendation system; user preferences; e-commerce platform

1. Introduction

In today's data-driven era, e-commerce platforms serve as a bridge connecting consumers with products. The precision of their marketing strategies directly impacts user experience, conversion rates, and market competitiveness [1]. With the rapid development of big data, artificial intelligence, and other technologies, e-commerce platforms have witnessed unprecedented marketing innovations [2]. Among these, precision marketing methods based on cluster analysis in e-commerce platforms calculate the distances between different data points to segment users into distinct groups or categories, thereby enabling personalized recommendations and coupon rewards [3-6]. However, this method primarily focuses on the application of cluster analysis techniques, with limited exploration of the integration with other data mining and machine learning technologies [7]. Research on the application of artificial intelligence technology in e-commerce platform marketing has provided more efficient and precise marketing strategies for e-commerce platforms through personalized recommendations, intelligent customer service, and market forecasting [8]. Intelligent recommendation systems analyze user behavior and preferences to provide personalized product and service recommendations, significantly enhancing user experience and purchase intent [9-11]. However, when discussing the advantages of artificial intelligence technology, there is insufficient attention to the potential data security and privacy protection issues it may raise, which is one of the significant challenges currently facing artificial intelligence technology in the marketing field [12].

Copyright: © 2026 by the authors

User profiling collects and analyzes multi-dimensional information such as users' basic attributes, behavioral trajectories, preferences, habits, and consumption capabilities to construct a three-dimensional, vivid user preference model. This helps e-commerce platforms establish an efficient and precise matching mechanism between a vast array of products and users, ensuring the accuracy and effectiveness of information [13-16]. Li, X proposed a method using artificial intelligence and big data analysis to construct accurate e-commerce user profiles. This method enhances personalized marketing and user satisfaction by precisely reflecting users' preferences and consumption characteristics [17]. Zhao, J et al. proposed a personalized recommendation system suitable for e-commerce platforms. This system combines social network user information with e-commerce product data and employs techniques such as topic modeling, feedback mechanisms, collaborative filtering, and clustering to achieve high recommendation accuracy [18]. Lv, S et al. integrated the concept of user nearest neighbors, proposing a novel hybrid e-commerce model. This algorithm uses a neighborhood model to mitigate the impact of data, optimizes to reduce complexity, and improves recommendation quality in the user-item interaction matrix [19]. Wang, L., and Jiang, Y. proposed an e-commerce collaborative recommendation method based on the fuzzy C-means clustering algorithm. This method analyzes user data using neural networks and data mining, then recommends products based on user interests and historical data [20]. Although the above studies have achieved significant results, recommendation algorithms still face some challenges, such as data sparsity, cold start problems, timeliness issues, and diversity recommendation problems.

In recent years, generative adversarial networks (GANs) have been widely adopted in personalized recommendation research due to their ability to generate realistic synthetic data that can effectively supplement real datasets [21]. The core of GANs is the game theory concept of a zero-sum game, where the gains of one participant necessarily imply losses for the other, and the total sum of gains and losses is always zero [22]. Zheng, T., et al. utilized the GAN framework to mine and fuse neighborhood information, combining explicit and implicit user information to construct an innovative method for horizontal recommendation systems, aiming to enhance the accuracy of Top-N recommendations [23]. Lu, X et al. proposed a recommendation model based on GAN for data compensation and dynamic user interest grouping to address data sparsity and user interest changes in e-commerce [24]. Ding, R et al. proposed a novel path-based adversarial recommendation model (path2rec) aimed at addressing the limitations of existing GAN-based methods by integrating auxiliary information, and demonstrated its effectiveness in product recommendation through experiments on real-world datasets [25].

Due to the challenges of training, poor controllability, and instability in the original GAN model, some scholars have proposed variants of GAN [26]. For example, Liu, Z., and Luo, W. proposed a GAN model named FMGAN, which employs a filter-enhanced multi-layer perceptron (MLP) generator and a linear discriminator to reduce bias and improve recommendation accuracy, and demonstrated its effectiveness on real-world datasets [27]. Chae, D. K., et al. combined deep learning and GAN techniques to propose a new recommendation framework called the Collaborative Adversarial Autoencoder (CAAE), which enhances recommendation accuracy by using an autoencoder as the generator, employing Bayesian personalized ranking, and generating negative items to improve recommendation accuracy [28]. Chonwiharnphan, P et al. proposed a personalized framework based on conditional generative adversarial networks (CGAN) to generate realistic user data for new products, which can be used for analysis to derive personalized schemes, and demonstrated its effectiveness in predicting new product features [29].

This paper proposes a personalized product recommendation method tailored to the diverse needs of users on e-commerce platforms, based on generative adversarial networks (GANs). The algorithm consists of two models: a generative model composed of multiple generators, each focusing on different product attributes, combined with domain knowledge to generate diverse product representations; and a discriminative model used to determine whether the generated products are real, providing feedback to continuously improve and stabilize the generative model. The algorithm was then tested using the ml-100k and ml-1m datasets, with comparative experiments conducted against other algorithms to validate its recommendation performance. Finally, the algorithm was trained and analyzed using the MovieLens-100K dataset.

2. Generative Adversarial E-Commerce Platform for Accurate Recommendations Based on User Preferences

2.1. Recommendation Systems and Generative Adversarial Networks

2.1.1. Recommendation System

Recommendation systems typically predict users' preferences for unknown items based on historical interaction information (browsing, playing, purchasing, etc.), user attribute information (age, gender, income, etc.), and item information (price, year of production, place of production, etc.), combined with

specific recommendation algorithms. The expression is shown in Formula (1):

$$\forall u \in U, i_u^{'} = \arg\max_{i \in I} s(u,i) \tag{1}$$

In this context, U (User) and I (Item) represent the sets of all users and all items in the recommendation system, respectively. s is a utility function used to calculate the recommendation score R of an item for a user, i.e., $s: U \times I \to R$. The task of the recommendation algorithm is to recommend the item $i_u^{'} \in I$ that a user is most likely to be interested in based on R. The recommendation algorithm needs to generalize the utility function s defined on the subspace $U \times I$ to the entire space $U \times I$. For example, in a rating prediction system, R is typically defined as the user's rating. However, in practice, the items that users have rated account for only a small portion of the total items, so when calculating the recommendation score of an item for a user, it is necessary to predict the unknown ratings based on the known ratings.

The input information for a recommendation system can be divided into explicit feedback information and implicit feedback information. The former typically provides a specific score from 1 to 5, while the latter typically includes browsing information, click information, and collection information, which only indicate the interaction between users and items and are generally represented in binary form as 0 or 1. However, in real-world scenarios, explicit feedback is often difficult to obtain because most users only engage in interaction behavior but rarely actively provide ratings. Implicit feedback is relatively easier to obtain and can succinctly express whether a user likes a particular item. Therefore, the work studied in this paper is also based on implicit feedback.

2.1.2. Generative Adversarial Networks

The generative adversarial network consists of two parts, namely generator G and discriminator D. where G is responsible for learning the distribution of real data and generating false data that is as similar as possible to real data [30]. D is a binary classifier, the goal of judging whether the input data is from real data G is to learn the distribution of real data $P_{data}(x)$, the input data is the random variable z (generally z is called noise), the output data is false data $G(z)$, and the data distribution learned by G is generally recorded as $P_z(x)$. The goal of D is to determine the probability that the input data is real data, and the closer the probability value is to 0, the greater the probability that D will think that the data is false. The closer it is to 1, the higher the probability that D will consider the data to be real. After the model is trained to the optimum, the fake data generated by G has the effect of being fake, and the output value of D is close to 0.5, that is, D is completely unable to distinguish whether the input data is the fake data generated by G or the real data, and it is generally believed that the false data is very similar to the real data in this case, and G has successfully learned the distribution of the real data.

The overall optimization objective of GAN is actually a minimization-maximization problem, with its objective function shown in Formula (2).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2}$$

The optimization of GANs is divided into two steps. In the first step, the parameters of G are not updated; only D is optimized to find the current optimal discriminator that can successfully distinguish between real and fake data. Correspondingly, in the second step, the parameters of D are not updated; only G is optimized to find the current optimal generator that produces data that is difficult for the discriminator to distinguish. In each optimization round, the discriminator should be slightly stronger than the generator, but not too much stronger. Therefore, in general, the discriminator is updated multiple times before the generator is updated once. G and D engage in a competitive game, ultimately reaching their optimal states.

However, the GAN generation process is not constrained, leading to uncontrollable generated data. The difference between CGAN and GAN lies in the addition of an extra conditional variable y to the inputs of both G and D. y can be a label, text, image, etc. CGAN uses the conditional variable y to guide and constrain the generation process, ensuring that the results generated by G fit the conditional variable y. If y is a category label, CGAN can be viewed as an improvement of the GAN model toward supervised learning.

The objective function of CGAN is similar to that of GAN, except that a conditional variable y is added to the input of G and D. The form of the objective function is shown in formula (3).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x \mid y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z \mid y)))] \tag{3}$$

The training steps for CGAN are similar to those for GAN. In equation (3), although the input parts of the variables $G(z \mid y)$ and $D(x \mid y)$ are written in the form of conditional probabilities, in the specific implementation of CGAN, it is only necessary to concatenate z or x with y.

2.2. User Preference Modeling

2.2.1. User Preference Map

First, construct a user preference graph $G = \{V, E, A\}$ based on the input user-clothing interaction sequence M^u. Each vertex $v \in V$ and $|V| = N_M$ corresponds to an interaction item, whose embedded representation is $\vec{h}_i$. E is the set of graph edges. $A \in \mathbb{R}^{N_M \times N_M}$ is the adjacency matrix of the graph, and $A_{i,j} \in E$ is the adjacency matrix, where $A_{i,j}$ indicates whether nodes i and j are related.

After reconstructing the user-clothing interaction sequence into a user preference graph, the more edges connecting the nodes, the stronger the correlation between the nodes, and the more prominent the user preferences of that node, i.e., the degree of the node reflects its importance. The more similar preferences there are, the more subgraphs there are in the user preference graph, and the denser the subgraphs are, the more users are interested in that type of preference. Since it is necessary to measure the similarity between adjacent nodes, the weighted cosine similarity $Sim_{ij} = \cos(\vec{w} \odot \vec{h}_i, \vec{w} \odot \vec{h}_i)$ is used as the metric function, where $\odot$ is the Hadamard product, $\vec{w}$ is a trainable weight vector used to adaptively highlight different dimensions of item embeddings. Considering computational complexity and node importance, non-negative elements in the similarity matrix are retained as follows:

$$A_{i,j} = \begin{cases} 1, Sim_{ij} \geq Rank_{\varepsilon N_M^2}(Sim) \\ 0, otherwise; \end{cases} \tag{4}$$

Among them, $Rank_{\varepsilon N^2}(Sim)$ returns the N_M^2 th largest value in matrix $A_{i,j}$, where ε controls the overall sparsity of the generated graph.

Then, the user preference graph divides the core preferences and interference terms into large clusters and small clusters. In the subsequent graph convolution process, information needs to be aggregated in the graph. By calculating the attention score for the target node h_i embedding and its neighborhood point set embedding, $\alpha_i = Attention_c(W_c\vec{h}_i \,||\, \vec{h}_{ic} \,||\, W_c\vec{h}_i \odot \vec{h}_{ic})$ where Attention is a two-layer feedforward neural network with LeakyReLU as the activation function, $\vec{h}_i$ is the embedding vector of a single node and $\vec{h}_i \in \mathbb{R}^{''}$, $\vec{h}_{ic}$ is the average embedding of the neighborhood point set, W_c is a trainable matrix, $||$ is the concatenated vector; then, the attention score $\beta_i = Attention_q(W_q\vec{h}_i \,||\, \vec{h}_t \,||\, W_q\vec{h}_i \odot \vec{h}_i)$ is calculated based on the correlation between the target node and the source node, where $\vec{h}_i$ is the source node embedding, $\vec{h}_i$ is the target node embedding, and W_q is a trainable matrix. Based on the two attention scores, the normalized attention coefficients are calculated using the softmax function:

$$E_{i,j} = softmax_j(\alpha_i + \beta_j) = \frac{\exp(\alpha_i + \beta_j)}{\sum_{k \in V} \exp(\alpha_i + \beta_k)} \tag{5}$$

Among these, α_i controls the target node receiving information, β_i controls the node sending information, and β_k is the attention score for the correlation between other target nodes in the point set v and the source node. Based on the normalized attention coefficients described above, a new node

embedding matrix $\{\vec{h}_1^{'}, \vec{h}_2^{'}, \ldots, \vec{h}_n^{'}\}$ is obtained, where $\vec{h}_i^{'} \in \mathbb{R}^n$ The node embedding vector $\vec{h}_i^{'} = \sigma(W_a \cdot sum(E_{i,j} \times \vec{h}_i \mid\mid j \in V) + \vec{h}_i)$ is defined, where σ is a nonlinear function and W_a is a trainable matrix.

Finally, the matrix $Q_i = softmax(W_p \cdot sum(A_{ij} \times \vec{h}_i^{'} \mid j \in V))$ is obtained through GNN pooling, where W_p is the weight matrix, the adjacency matrix of the pooled graph $A^* = Q_i^T AQ$, and its node embedding matrix $\{\vec{h}_1^*, \vec{h}_2^*, \ldots, \vec{h}_m^*\} = Q_i^T \{\vec{h}_1^{'}, \vec{h}_2^{'}, \ldots, \vec{h}_n^{'}\}$, and the importance scores are $\{\gamma_1^*, \gamma_2^*, \ldots, \gamma_m^*\} = Q_i^T \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$. Considering that the temporal relationships between different clusters in the user preference graph are difficult to determine, relative position regularization is applied to obtain the user core preference sequence $M^u c = \{i_1^u, i_2^u, \ldots, i_{N_{M_{uc}}^u}\}$.

2.2.2. Users' Long-Term and Short-Term Preferences

For the current user's core preference sequence, considering the complexity of the method, this section uses two RNN networks with shared weights [31] to extract the user's long-term and short-term preferences. Users' long-term preferences require modeling the entire core preference sequence, and the RNN sequence structure aligns well with the characteristics of user interaction sequences. Additionally, user-clothing long-term preferences are relatively insensitive to time. Considering that user preferences may undergo non-immediate changes, directly modeling the entire user-clothing interaction sequence when constructing user-clothing long-term preferences can generally meet this requirement. Therefore, this section uses an RNN network to recursively process all items in the user's core preference sequence, traversing the entire sequence to obtain the user-clothing long-term preference $S_u^l = RNN(M^{uc}[i_1^u : i_{N_{M_u}}^u])$. To capture users' rapidly changing preferences, it is necessary to simultaneously understand users' immediate preferences, which requires modeling the final interaction items in the user-clothing interaction sequence. Since user-item short-term preferences are time-sensitive, interactions between adjacent items more accurately reflect user-item short-term preferences. Therefore, when constructing user-item short-term preferences, the final item must be considered for inference. Thus, the aforementioned RNN network with shared weights is used to model user-item short-term preferences $S_u^s = RNN(M^{uc}[i_{N_{M_u}}^u])$.

2.3. GAN Recommendation Algorithm Based on User Preferences and Data Augmentation

2.3.1. Modeling Diverse Users

To make targeted recommendations, we first need to figure out if the current user is diverse. The process is shown in Figure 1. First, we get project info from the user's interaction history, then we get the category attributes for the project. The user diversity tendency ω is calculated as per Formula (6). If $\omega < P$, the GRU-NR model (GRU-NR) is employed to ensure the accuracy of the recommendation results. If $\omega > P$, the user is classified as having diverse needs, and the UR-GAN algorithm (UR-GAN) is used for recommendations to meet the user's diverse needs.

$$\omega = \frac{L}{K} \tag{6}$$

Among them, L is the number of categories of all interaction items of users, K is the total number of interaction items of users, and $L \le K$.

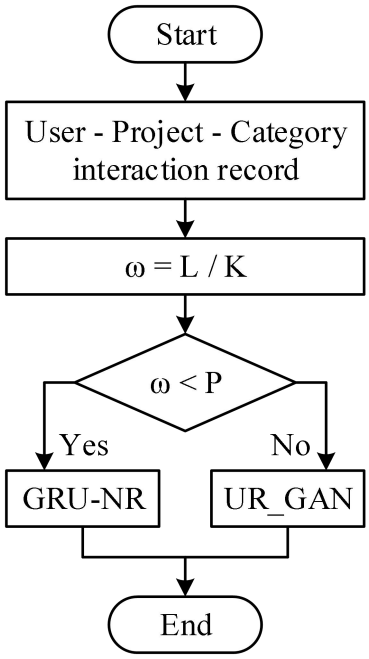


Figure 1. Diversity user decision flowchart.

2.3.2. UR-GAN Recommendation Algorithm

The structure of the generative adversarial recommendation algorithm based on user needs is shown in Figure 2. It consists of two parts. The generative model aims to recommend diversity and generates project representations that meet user needs. The discriminative model identifies real projects and generated projects and provides feedback to the generative model, prompting the generative model to generate projects that are infinitely close to real projects and sufficient to deceive the discriminative model.

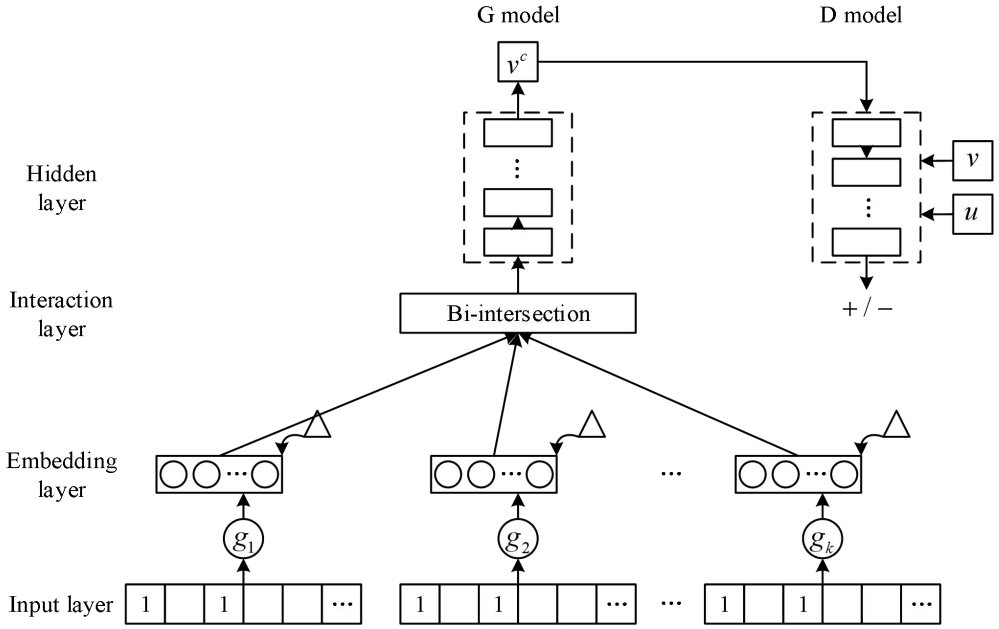


Figure 2. Structure diagram of UR-GAN algorithm.

(1) Generative Model

The input to the generative model is the attribute information of user-interaction items, i.e., user-attribute interactions, and the output is a diverse representation of items based on user needs. This representation is generated from multiple attribute aspects and fully reflects users' diverse needs.

The input to the input layer is a user-attribute interaction vector, where each vector represents a different attribute. In each interaction vector, 1 indicates that the user has interacted with a specific value of the attribute, and 0 indicates that the user has not interacted with a specific value of the attribute. Each attribute has multiple specific values, as shown in Formula (7), where l denotes the number of specific values, determined by the specific item and its attributes. For example, the brand attribute of a car has multiple specific values such as BMW, Mercedes-Benz, and Audi.

$$c_{ij} = \{c_{ij}^1, c_{ij}^2, ...c_{ij}^l\} \tag{7}$$

The embedding layer consists of multiple generators, each corresponding to a project attribute, with each generator aiming to achieve diversity in recommendations, ensuring that recommended projects cover a wider range of different attributes. This allows each generator to generate potential project features from different attribute perspectives. Additionally, considering the need to integrate domain knowledge and user requirements, the idea introduced is that each attribute contains different specific values, and each specific value should not be treated equally. For example, different weights should be assigned to different users and scenarios to distinguish the importance of feature values. Finally, the project features generated by each generator based on different attributes are combined to form the overall features of the user's preferred projects, as shown in Formula (8).

$$e = \{g_1(c_{i1}), g_2(c_{i2}), ...g_k(c_{ik})\} \tag{8}$$

In this context, g represents the generator, and k denotes the number of generators, which aligns with the number of project attributes.

The interaction layer establishes relationships between attributes, as the generated project features are composed of different attributes that are not independent but interconnected. For instance, there is a connection between brand attributes and price attributes, with high-end brand products typically priced slightly higher than mass-market brand products. Interaction operations are shown in formula (9), where $\otimes$ represents element-level interaction, i.e., the corresponding elements are multiplied, and x_i, x_j are second-order features. The final result is the features obtained after all attributes are interacted with each other in pairs.

$$v^{BI} = f_{BI}(e) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} e_i x_i \otimes e_j x_j \tag{9}$$

The hidden layer consists of multiple fully connected layers, which take the cross-product results obtained from the interaction layer as input and perform deep interactions between first-order and second-order features, as shown in Formula (10), where f_L^g represents the activation function of the fully connected layer, z_L^g is the output of each layer, and W_L^g and b_L^g denote the weight and bias terms of the fully connected network. The final generated project representation is composed of attribute features, and the attributes undergo deep interaction between features.

$$\begin{aligned} z_1^g &= v^{BI} \\ z_2^g &= f_2^g(W_2^g z_1^g + b_2^g) \\ &... \\ v^c &= f_L^g(W_L^g z_{L-1}^g + b_L^g) \end{aligned} \tag{10}$$

(2) Discriminative Model

In generative adversarial networks, in order to train a more robust generator, the discriminator needs to continuously promote the iterative training of the generator. The discriminative model attempts to distinguish between generated items and real items, and feeds the results back to the generative model, prompting the generative model to generate items that are closer to real items.

The discriminator model takes real items, generated items, and corresponding users as input, and outputs the matching degree between users and items, denoted as $D(v \mid u_m)$ as shown in Formula (11).

$$D(v \mid u_m) = \sigma(d_\phi(v \mid u_m)) = \frac{\exp(d_\phi(v \mid u_m))}{1 + \exp(d_\phi(v \mid u_m))} \tag{11}$$

where σ is the sigmoid function, used to calculate the probability of matching users with projects. d_ϕ is the discriminative model, consistent with the hidden layer of the generative model, composed of multiple fully connected layers, as shown in formula (12).

$$\begin{aligned} z_1^d &= v \oplus u_m \\ z_2^d &= f_2^d(W_2^d z_1^d + b_2^d) \\ &\cdots \\ d_\phi(v \mid u_m) &= f_L^d(W_L^d z_{L-1}^d + b_L^d) \end{aligned} \tag{12}$$

Concatenate the user vector and project vector as the input to the fully connected layer. Similarly, f_L^d represents the activation function of the fully connected layer, z_L^d is the output of each layer, and W_L^d and b_L^d represent the weight and bias terms of the fully connected network.

2.3.3. Model Training

Generative adversarial training involves a game of maximization and minimization between the generator and the discriminator, which mutually improve each other, ultimately enabling the generator to deceive the discriminator. The generative model takes user-attribute records as input and aims to achieve diversity-based recommendations. Each attribute corresponds to a generator, which generates project representations from different attribute perspectives. The discriminative model distinguishes between real projects and generated projects, prompting the generative model to generate projects that are closer to reality and better matched to the user. In summary, the overall objective function of the algorithm is as shown in Formula (13).

$$J^{G^*,D^*} = \min_\theta \max_\phi \sum_{u=1}^{m} (E_{v \sim P_{reas}(v|\mu_m)}[\log(D(v \mid u_m))] + E_{v^c \sim P_0(v^c|\mu_m)}[\log(1 - D(v^c \mid u_m))]) \tag{13}$$

Among them, $P_{true}(v \mid u_m)$ is the probability distribution of the true item v, and $P_0(v^c \mid u_m)$ is the probability distribution of the generated item v^c.

(1) Optimizing the generative model

The generative model can learn to generate item representations based on given user and item attribute information, with the goal of diversity-based recommendation. The generative model minimizes the overall objective function to ensure that the difference between the generated items and the true items is sufficiently small, thereby deceiving the discriminator and making it difficult to distinguish between true items and generated items. This is illustrated in formula (14).

$$\begin{aligned} &\min_\theta \sum_{u=1} (E_{v^c \sim P_0(v^c|u_m)}[\log(1 - 8(u_\phi(v \mid u_m)))]) \\ &= \max_\theta \sum_{u=1}^{m} (E_{v^c \sim P_0(v^c|u_m)}[\log(1 + \exp(d_\phi(v^c \mid u_m)))]) \end{aligned} \tag{14}$$

(2) Optimizing the discriminative model

Unlike generative models, discriminative models aim to maximize the overall objective function and clearly distinguish between real projects and generated projects. The evaluation results are then fed back to the generative model to prompt it to train and adjust its parameters. Through continuous iterative optimization, the best project representation is ultimately generated. See formula (15) for details.

$$\max_\phi \sum_{u=1}^{m} (E_{v \sim P_{pac}(v|u_m)}[\log(\sigma(d_\phi(v \mid u_m)))] + E_{v^c \sim P_0(v \mid u_m)}[\log(1 - \sigma(d_\phi(v^c \mid u_m)))]) \tag{15}$$

3. Experimental Results and Analysis

3.1. Experimental Setup

3.1.1. Experimental Setup and Dataset

The algorithm in this paper runs on a Windows 10 operating system, Intel Core i7 CPU, 16 GB of memory, and Nvidia GeForce GTX 1080 graphics card. The ml-100k and ml-lm datasets from the publicly available Movie Lens dataset were selected for experimentation, and the dataset was randomly divided into a training set and a test set at an 8:2 ratio. The training process used the Adam optimizer from the stochastic gradient descent method [32]. This paper considers implicit feedback, where if item i is relevant (i.e., user u has a rating record for item i), $r_{wi}=1$, otherwise $r_{ui}=0$. Table 1 provides the statistical information for these two datasets.

After multiple experimental comparisons, the neural network hidden layer was set to three layers, the discriminator network hidden layer node count was set to {1024, 128, 16}, and the generator network hidden layer node count was set to {256, 512, 1024}. Additionally, some empirically well-performing hyperparameters were determined, with the sigmoid function used as the activation function for the neural network. with a learning rate of 0.0001 and a batch size of 32. We varied S^{ZR} and S^{PM} using {10, 30, 50, 70, 90} and the regularization coefficient α using {0.5, 0.25, 0.1, 0.05, 0.01} to compare the impact of these hyperparameters on recommendation accuracy.

Table 1. Statistics of data sets.

Data set	Number of users	Project number	Score number	Sparse degree /%
ml-100k	916	1457	10000	91.09
ml-1m	5593	3472	100175	94.54

3.1.2. Evaluation Indicators

This paper employs four commonly used accuracy metrics for the recommendation task: precision (P@N), recall (R@N), normalized discounted cumulative gain (NDCG@N), and mean reciprocal rank (MRR@N). The first two metrics focus on how many correct items are included in the recommendation list, while the latter two metrics focus on the ranking positions of correct items within the recommendation list. For a user u, if N items are recommended (denoted as $R(u)$), and the set of items interacted with by user u in the test set is $T(u)$, then the formulas for precision and recall are as follows:

$$P@N=\frac{\sum_{u}|R(u)\cap T(u)|}{\sum_{u}|R(u)|} \tag{16}$$

$$R@N=\frac{\sum_{u}|R(u)\cap T(u)|}{\sum_{u}|T(u)|} \tag{17}$$

NDCG@N can be expressed as:

$$NDCG@N=\frac{\sum_{u}NDCG_{u}@N}{|U|} \tag{18}$$

The MRR@N formula is:

$$MRR@N=\frac{1}{|U|}\sum_{\alpha}\frac{1}{rank_{\alpha}} \tag{19}$$

Among them, $|U|$ is the number of users in the test set, and $rank_u$ represents the position of the first correct recommendation in the recommendation list of user u.

The larger the above four evaluation indicators are, the better the recommendation performance of the model. In this paper, the recommendation list length N is fixed at 5 and 20.

3.2. Algorithm Testing and Evaluation

3.2.1. Algorithm Execution Efficiency Test

For algorithm efficiency testing, different algorithms were used for recommendation testing. The

tests were conducted using data volumes of 100,000 and 1,000,000, respectively, with 100 tests performed for each, and the average time was calculated. The algorithms were executed using only the CPU, and the test results are shown in Table 2. As shown in the table, for the GAN algorithm, the recommendation efficiency decreases significantly as the data volume increases. However, the PD-GAN algorithm effectively accelerates the recommendation efficiency, improving it by approximately 8 times. Although the user-demand-based generative adversarial recommendation algorithm requires a relatively long training time initially, once the model is trained, subsequent recommendations can be made with a single forward pass through the network, consuming minimal time. This significantly enhances the user experience.

Table 2. Time-consuming algorithm.

Algorithm name	100,000 data takes time(second)	A million data takes time(second)
GAN	2.7	49.5
PD-GAN	2.2	8.3
DRMUD	0.15	0.15
UR-GAN	0.15	0.15

3.2.2. Data Volume Requirements Assessment

Traditional GAN algorithms and the UR-GAN recommendation algorithm can achieve certain recommendation effectiveness with a sufficient amount of data. However, the amount of data required for deep learning algorithms is extremely large. The recommendation accuracy rates for the top 1 recommendations were evaluated under different data volumes of 10,000, 100,000, and 1,000,000 training data points, with the detailed results shown in Table 3.

The data in the table leads to the conclusion that as the data volume increases from 10,000 to 100,000, the accuracy improvement of GAN and PD-GAN algorithms becomes minimal. Once the data volume exceeds 100,000, the accuracy improvement of GAN and PD-GAN algorithms has almost ceased. However, for a data volume of 10,000, the accuracy of the DRMUD recommendation algorithm is very low and does not match the recommendation performance of traditional recommendation algorithms. However, when the data volume exceeds 100,000, the accuracy of DRMUD shows a significant improvement and significantly outperforms traditional recommendation algorithms. Additionally, it is evident that due to the presence of the GAN neural network, the training accuracy of the UR-GAN recommendation algorithm far exceeds that of the standalone generative adversarial network when the data volume reaches 100,000.

Table 3. Training data required by the algorithm.

Algorithm name	10,000 data	100,000 data	Million data
GAN	68.4%	79.3%	81.7%
PD-GAN	66.8%	75.2%	83.4%
DRMUD	46.3%	60.4%	86.7%
UR-GAN	61.5%	75.1%	88.9%

3.3. Comparative Experiments

This section compares the recommendation performance of the proposed method with that of commonly used Top-N recommendation algorithms to verify whether the proposed method can improve recommendation accuracy. The specific comparison methods are as follows:

ItemPop: The simplest non-personalized recommendation algorithm, which ranks items in descending order of popularity (i.e., the number of purchase records).

MF-BPR: A matrix factorization-based ranking algorithm that optimizes rankings based on each user's individual item preferences, requiring a training set of user preference triples for items.

IRGAN: Uses conditional generative adversarial networks to generate discrete item indices that mimic users' real interactions.

CFGAN-I: Uses vector-based conditional adversarial networks to learn the distribution of real data, with the learning condition being users' real interactions.

Figure 3 shows the performance comparison between user-based and user interaction vector-based methods on the ml-100k dataset, with the number of iterations set to 1000. First, as shown in the figure, the UR-GAN recommendation method based on user preferences generally outperforms the recommendation method based on user interaction vectors across all evaluation metrics. This overcomes the ambiguity of unknown items in the CFGAN-type method based on user interaction vectors, which

introduces errors in the fully connected layer of the input neural network, thereby validating the effectiveness of the proposed method. Second, due to the difficulty of convergence in zero-sum games in GAN models, they are susceptible to hyperparameter influences. As shown in the figure, the evaluation metrics of the CFGAN algorithm based on user interaction generally show a trend of first increasing and then decreasing. It begins to converge after approximately 600 iterations, but as the number of iterations increases, its evaluation metrics begin to decline. This is because the user interaction vector exhibits overfitting, leading to a decline in recommendation accuracy. In contrast, the evaluation metrics of the UR-GAN recommendation model based on user needs remain stable after convergence. Based on the above analysis, it can be concluded that the UR-GAN recommendation model based on user needs exhibits smoother and more stable convergence in recommendation accuracy, thereby validating its superior convergence performance.

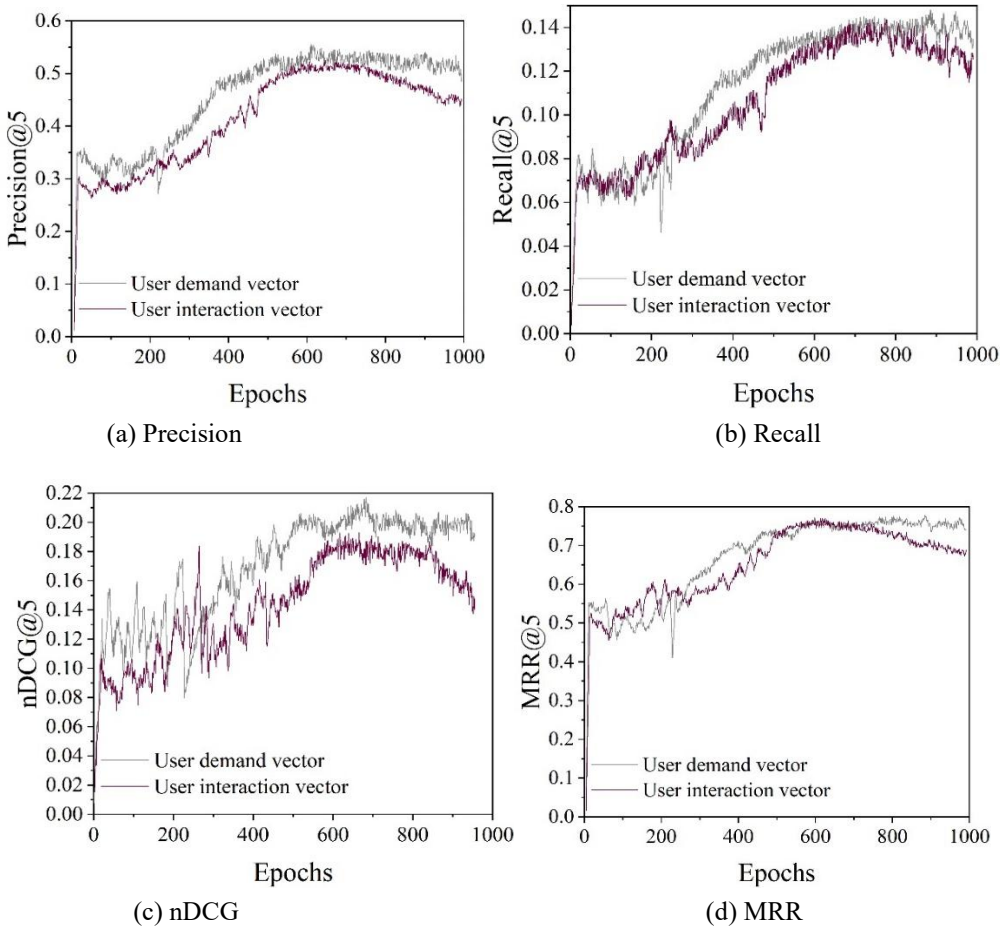


Figure 3. Two types of input conditions can be compared.

Tables 4 and 5 further compare the metric performance of premium and regular products recommended by the two datasets on e-commerce platforms under different recommendation methods, with P, R, G, and M representing Precision, Recall, NDCG, and MRR, respectively.

It can be seen from Table 4 that UR-GAN based on user needs is slightly better than other recommendation methods in each evaluation index, and the recommendation accuracy is 2.7 times that of non-personalized recommendation ItemPop p@5, which is 0.08 times higher than that of CFGAN-I based on user interaction vector, and 0.01 r@5 higher than that of CFGAN-I. In the recommended position ranking metric G@20 it is twice as high as ItemPop, with an increase of 0.03 compared to CFGAN-I and a m@5 improvement of 0.08 compared to CFGAN-I. Compared with other methods, CFGAN-I also has a certain improvement in accuracy when recommending products, but the improvement effect is slightly reduced compared with ordinary products because it recommends products that are not highly correlated. Table 5 also shows that the proposed method is superior to other recommendation methods, which verifies that the UR-GAN recommendation model based on user needs can improve the recommendation performance.

Table 4. The ml-100k data set different recommendations for performance comparison.

Recommendation algorithm	P@5	R@5	G@5	M@5
ItemPop	0.2272	0.0981	0.1024	0.4732
MF-BPR	0.2153	0.1026	0.1357	0.5461
CFGAN-I	0.4971	0.1124	0.1679	0.6974
UR-GAN	0.5345	0.1592	0.1893	0.7784
Recommendation algorithm	P@20	R@20	G@20	M@20
ItemPop	0.1783	0.1945	0.1811	0.4529
MF-BPR	0.1455	0.2174	0.1921	0.5318
CFGAN-I	0.3764	0.2946	0.3058	0.6671
UR-GAN	0.3985	0.2913	0.3347	0.7593

Table 5. The performance comparison of different recommendation methods for ml-1m data sets.

Recommendation algorithm	P@5	R@5	G@5	M@5
ItemPop	0.2272	0.0981	0.1024	0.4732
MF-BPR	0.2153	0.1026	0.1357	0.5461
CFGAN-I	0.4971	0.1124	0.1679	0.6974
UR-GAN	0.5345	0.1592	0.1893	0.7784
Recommendation algorithm	P@20	R@20	G@20	M@20
ItemPop	0.1783	0.1945	0.1811	0.4529
MF-BPR	0.1455	0.2174	0.1921	0.5318
CFGAN-I	0.3764	0.2946	0.3058	0.6671
UR-GAN	0.3985	0.2913	0.3347	0.7593

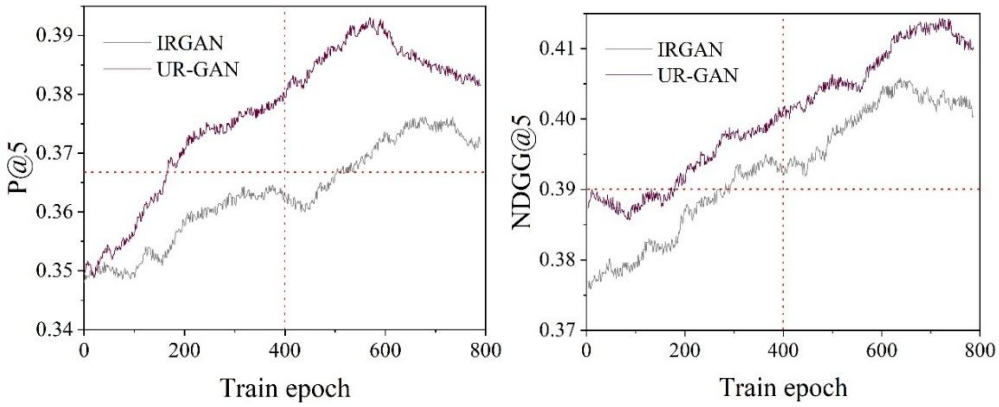
3.4. Analysis of the Model Training Process

This section provides an in-depth analysis of the model training process to draw some important conclusions. We compare the training process curves (for generative models) of the IRGAN model and the UR-GAN model on the Movie-Lens-100K dataset, focusing on the P@5 and NDCG@5 metrics (other metrics are similar), as shown in Figure 4. The results for the Movie-Lens-1M dataset are similar.

For IRGAN, when the number of iterations reaches around 650, the recommendation performance of IRGAN reaches its optimal level, indicating that the model has converged. Continuing to increase the number of iterations results in a slight decline in recommendation performance. For UR-GAN, when the number of iterations reaches around 550, the recommendation performance of UR-GAN reaches its optimal level, indicating that the model has converged. Clearly, UR-GAN achieves faster convergence than IRGAN, and this advantage becomes more pronounced when training on large datasets.

In the figure, the optimal performance of IRGAN is P@5=0.3753 and NDCG@5=0.4011, while the optimal performance of UPM-GAN is P@5=0.3908 and NDCG@5=0.4133, respectively. UR-GAN recommends better performance. The results show that UR-GAN has obvious advantages in improving accuracy, better mining users' implied preferences, and then accurately predicting scores and improving recommendation performance.

The training process curve of UR-GAN is smoother than that of IRGAN, meaning there are fewer instances of meandering, abrupt changes, and spikes. As a result, the model training process is more stable, with no significant “bumpy” phenomena observed. The process is more stable, with no significant “jumps” occurring. Due to UR-GAN's fast convergence speed, the training cost is significantly reduced. Therefore, the UR-GAN model not only offers superior recommendation performance but also has excellent real-time runtime efficiency, making it more practical for recommendation systems.

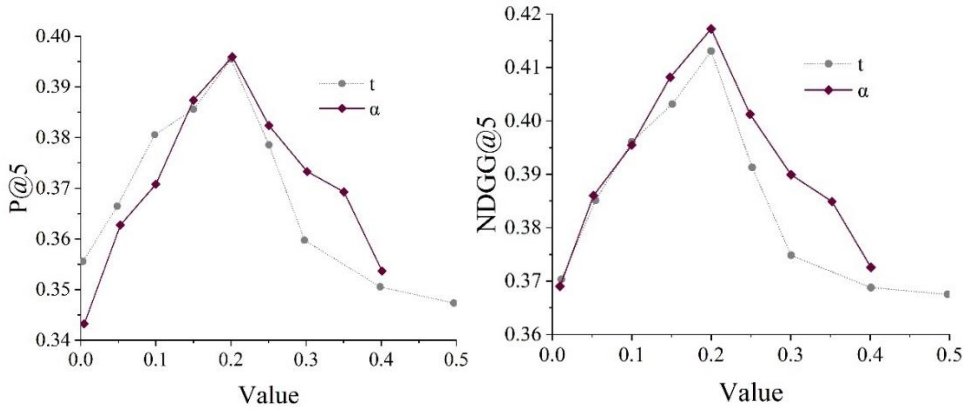


(a) Comparison 1 (b) Comparison 2

Figure 4. Learning curves of different recommendation systems.

Figure 4 shows the dynamic adjustment process of the t parameter in UR-GAN on the MovieLens-100K dataset, with P@5 and NDCG@5 selected as the metrics for display. Correspondingly, Figure 5 shows the adjustment process of the α parameter in UR-GAN on the MovieLens-100K dataset, with P@5 and NDCG@5 also selected as the metrics for display.

In the figure, it can be seen from the dynamic adjustment process of the t parameter in the generative model that the trends of the P@5 and NDCG@5 metrics are very similar. When t > 0, the recommendation performance of UR-GAN gradually increases until t = 0.2, at which point the recommendation performance is optimal. After that, as t increases, the recommendation performance declines. This indicates that in UR-GAN, the generative model must select an appropriate sensitivity parameter t based on the probability distribution of users and projects to achieve optimal recommendations. When α > 0, the recommendation performance of UR-GAN gradually improves until α = 0.20, at which point the algorithm obtains the most reasonable sample distribution, thereby optimizing the performance of the recommendation system.



(a) P@5 (b) NDGG@5

Figure 5. Performance variations of P@5 and NDCG@5 with respect to α.

4. Conclusion

This paper proposes a generative adversarial recommendation algorithm based on user needs. For users with diverse needs, accuracy-based recommendations are employed to ensure that the recommendation list closely aligns with user preferences, thereby enabling product recommendations on e-commerce platforms. Experimental results indicate that when the dataset is small, traditional GAN algorithms exhibit faster execution speeds and maintain decent accuracy rates. However, as the dataset grows to a certain scale, the advantages of the proposed recommendation model become increasingly evident. Furthermore, the user-demand-based generative adversarial recommendation algorithm outperforms other recommendation algorithms, validating that this method can improve recommendation accuracy. According to the model training results, the user-demand-based generative adversarial recommendation algorithm not only demonstrates excellent recommendation performance

but also converges faster and undergoes a smoother training process. Therefore, the practical value of the user-demand-based generative adversarial recommendation algorithm for recommendation systems is relatively high.

References

1. Zhang, Z., Xu, G., & Zhang, P. (2016). Research on E-Commerce Platform-Based Personalized Recommendation Algorithm. Applied computational intelligence and soft computing, 2016(1), 5160460.
2. Xu, L., & Sang, X. (2022). E-Commerce Online Shopping Platform Recommendation Model Based on Integrated Personalized Recommendation. Scientific Programming, 2022(1), 4823828.
3. Zhang, B., Wang, L., & Li, Y. (2021). Precision Marketing Method of E-Commerce Platform Based on Clustering Algorithm. Complexity, 2021(1), 5538677.
4. Yang, B., & Li, J. (2022). Precise Marketing Strategy Optimization of E-Commerce Platform Based on KNN Clustering. Journal of Mathematics, 2022(1), 7957509.
5. Zhang, D., & Huang, M. (2022). A Precision Marketing Strategy of e-Commerce Platform Based on Consumer Behavior Analysis in the Era of Big Data. Mathematical Problems in Engineering, 2022(1), 8580561.
6. Guo, S., & Zhai, R. (2022). E-commerce precision marketing and consumer behavior models based on IoT clustering algorithm. Journal of Cases on Information Technology (JCIT), 24(5), 1-21.
7. Song, X., Yang, S., Huang, Z., & Huang, T. (2019, August). The application of artificial intelligence in electronic commerce. In Journal of Physics: Conference Series (Vol. 1302, No. 3, p. 032030). IOP Publishing.
8. Jawaheer, G., Weller, P., & Kostkova, P. (2014). Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. ACM Transactions on Interactive Intelligent Systems (TiiS), 4(2), 1-26.
9. Gemmis, M. D., Iaquinta, L., Lops, P., Musto, C., Narducci, F., & Semeraro, G. (2010). Learning preference models in recommender systems. In Preference Learning (pp. 387-407). Berlin, Heidelberg: Springer Berlin Heidelberg.
10. Yu, Z., Lian, J., Mahmoody, A., Liu, G., & Xie, X. (2019, August). Adaptive User Modeling with Long and Short-Term Preferences for Personalized Recommendation. In IJCAI (Vol. 7, pp. 4213-4219).
11. Xu, K., Zhou, H., Zheng, H., Zhu, M., & Xin, Q. (2024). Intelligent classification and personalized recommendation of e-commerce products based on machine learning. arXiv preprint arXiv:2403.19345.
12. Li, C. (2024). A Personalized Product Recommendation System for E-Commerce Platforms Based on Artificial Intelligence and Image Processing Technologies. Traitement du Signal, 41(6).
13. Wu, T., Yang, F., Zhang, D., Zhu, A., & Wan, F. (2020, March). Research on Recommendation system based on user portrait. In 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS) (pp. 462-465). IEEE.
14. Yao, W., Hou, Q., Wang, J., Lin, H., Li, X., & Wang, X. (2019, July). A personalized recommendation system based on user portrait. In Proceedings of the 2019 international conference on artificial intelligence and computer science (pp. 341-347).
15. Fang, H., Guo, G., & Zhang, J. (2015). Multi-faceted trust and distrust prediction for recommender systems. Decision Support Systems, 71, 37-47.
16. Li, L. (2022). Cross-Border E-Commerce Intelligent Information Recommendation System Based on Deep Learning. Computational intelligence and neuroscience, 2022(1), 6602471.
17. Li, X. (2025). An accurate construction method of E-commerce user profile based on artificial intelligence algorithm and big data analysis. International Journal of High Speed Electronics and Systems, 34(01), 2540107.
18. Zhao, J., Su, B., Rao, X., & Chen, Z. (2022). A cross-platform personalized recommender system for connecting e-commerce and social network. Future Internet, 15(1), 13.
19. Lv, S., Wang, J., Deng, F., & Yan, P. (2024). A hybrid recommendation algorithm based on user nearest neighbor model. Scientific Reports, 14(1), 17119.
20. Wang, L., & Jiang, Y. (2022). Collocating Recommendation Method for E-Commerce Based on Fuzzy C-Means Clustering Algorithm. Journal of Mathematics, 2022(1), 7414419.
21. Song, Y., Li, J., Fan, X., & Wang, Y. (2024, December). Research on E-commerce Personalized Visual Marketing Algorithms Based on Generative Adversarial Network (GAN) Model. In Proceedings of the 2024 7th International Conference on E-Business, Information Management and Computer Science (pp. 95-101).
22. Manaa, N., Seridi, H., & Mendjel, M. S. M. (2024). Advancements in Recommender Systems through the Integration of Generative Adversarial Networks. International Journal of Informatics and Applied Mathematics, 6(2), 35-45.
23. Zheng, T., Li, S., Liu, Y., Zhang, Z., & Jiang, M. (2024). An effective neighbor information mining and fusion method for recommender systems based on generative adversarial network. Expert Systems with Applications, 248, 123396.
24. Lu, X., Liu, J., Gan, S., Li, T., Xiao, Y., & Liu, Y. (2021). Recommendation model based on dynamic interest group identification and data compensation. IEEE Transactions on Network and Service Management, 19(1), 89-99.
25. Ding, R., Chen, B., Guo, G., & Yang, X. (2020). Adversarial path sampling for recommender systems. IEEE Intelligent Systems, 36(6), 23-31.

26. Chen, H., Wang, S., Jiang, N., Li, Z., Yan, N., & Shi, L. (2021). Trust-aware generative adversarial network with recurrent neural network for recommender systems. International Journal of Intelligent Systems, 36(2), 778-795.
27. Liu, Z., & Luo, W. (2023). FMGAN: A Filter-Enhanced MLP Debias Recommendation Model Based on Generative Adversarial Network. Applied Sciences, 13(13), 7975.
28. Chae, D. K., Shin, J. A., & Kim, S. W. (2019). Collaborative adversarial autoencoders: An effective collaborative filtering model under the GAN framework. IEEE Access, 7, 37650-37663.
29. Chonwiharnphan, P., Thienprapasith, P., & Chuangsuwanich, E. (2020). Generating realistic users using generative adversarial network with recommendation-based embedding. IEEE Access, 8, 41384-41393.
30. Jabbar Hussain,Magnus Båth & Jonas Ivarsson. (2025). Generative adversarial networks in medical image reconstruction: A systematic literature review. . Computers in biology and medicine,191,110094.
31. Alexey G. Zinyagin,Alexander V. Muntin,Vadim S. Tynchenko,Pavel I. Zhikharev,Nikita R. Borisenko & Ivan Malashin. (2024). Recurrent Neural Network (RNN)-Based Approach to Predict Mean Flow Stress in Industrial Rolling. Metals,14(12),1329-1329.
32. Mohamed Amine Mahjoubi,Driss Lamrani,Shawki Saleh,Wassima Moutaouakil,Asmae Ouhmida,Soufiane Hamida... & Abdelhadi Raihani. (2025). Optimizing ResNet50 performance using stochastic gradient descent on MRI images for Alzheimer's disease classification. Intelligence-Based Medicine,11,100219-100219.