STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, May 2026, pp 4417–4436.
Published online in International Academic Press (www.IAPress.org)

An adversarial framework with dual genetic optimization for similarity-aware matrix factorization in recommendation systems

Sanae Filali-Zegzouti[1,*], Oumayma Banouar [2], Mohamed Benslimane [1]

[1]*Laboratory for Science, Engineering and Management, Faculty of Science and Technology, USMBA, Fez, Morocco*
[2]*Laboratory of Applied Mathematics and Informatics, Faculty of Sciences and Technology, UCA, Marrakech, Morocco*

Abstract Modern recommendation systems (RS) continue to face critical challenges, including data sparsity and the difficulty of modeling complex user–item interactions, which hinder their ability to deliver accurate and personalized recommendations. To address these limitations, we propose GaSimGAN, a novel collaborative filtering framework that integrates Generative Adversarial Networks (GANs), similarity-aware modeling, and a dual genetic optimization strategy. The proposed framework leverages personalized similarity matrices to focus on the most relevant users and items, thereby refining the input space of the generative process and improving recommendation accuracy. GaSimGAN adopts a matrix factorization-based generator coupled with an autoencoder-based discriminator, enriched with Pearson similarity information to better capture relational patterns in user–item interactions.Genetic Algorithms are employed in a dual role: first, during preprocessing to optimize similarity-based neighbor selection, and second, as an offline one-shot hyperparameter optimization step conducted prior to and independently of the adversarial training pipeline. Extensive experiments conducted on benchmark datasets, including MovieLens 1M, HetRec2011, and LastFM, demonstrate that GaSimGAN consistently outperforms state-of-the-art GAN-based recommendation systems in terms of Precision, Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG), confirming both its effectiveness and scalability.

Keywords Recommendation systems, Generative Adversarial Networks, Personalization for recommendation, Dual Genetic Algorithms, Pearson Similarity, Model based Collaborative Filtering

AMS 2010 subject classifications 68T05, 68P20

DOI: 10.19139/soic-2310-5070-3413

1. Introduction

In an era of unprecedented information overload, recommendation systems (RS) have become essential infrastructure for delivering personalized access to products, services, and digital content. By analyzing historical interactions, item attributes, and behavioral patterns, RS fundamentally reshape how users discover and engage with information across diverse platforms [1, 2, 3].

The effectiveness of RS depends critically on user feedback, which manifests in two forms. Explicit feedback, including ratings, reviews, and direct preferences, provides unambiguous signals but remains chronically sparse. [4]. Implicit feedback, inferred from actions such as clicks, views, or purchase history, is abundant yet inherently ambiguous: it confirms interaction occurrence but not satisfaction [5, 6]. This ambiguity poses fundamental challenges for accurate preference modeling.

Among established RS paradigms, such as content-based filtering, collaborative filtering (CF), and hybrid methods [7], CF remains dominant due to its ability to discover latent patterns from collective user behavior. Model-based CF has evolved along two principal directions. Matrix factorization decomposes user-item interactions into

*Correspondence to: Sanae Filali-Zegzouti (Email: sanae.filalizegzouti@usmba.ac.ma). Laboratory for Science, Engineering and Management, Faculty of Science and Technology, USMBA, Fez, Morocco

ISSN 2310-5070 (online) ISSN 2311-004X (print)

low-dimensional latent representations [8, 9], offering computational efficiency and interpretability. However, MF's linear assumptions, sensitivity to sparsity, and static similarity modeling limit its expressiveness. Deep learning approaches, including autoencoders [10], variational autoencoders [11], and generative adversarial networks (GANs) such as IRGAN [12], capture complex nonlinearities and synthesize realistic user profiles to mitigate sparsity. Yet these methods face persistent challenges: unstable adversarial training, limited interpretability, and emerging ethical risks including bias amplification and item hallucination. [13, 14, 15].

Despite advances in both MF and GAN-based approaches, three fundamental limitations persist. First, MF models remain predominantly linear, constraining their capacity to capture intricate user-item relationships. Second, adversarial training in GAN-based recommenders suffers from instability and mode collapse, particularly under sparse data conditions. Third, similarity information, a cornerstone of collaborative filtering, is rarely integrated explicitly into generative frameworks, resulting in suboptimal personalization.

To address these challenges, we propose *GaSimGAN* (Genetic Algorithm-enhanced Similarity-aware Generative Adversarial Network), a novel CF framework that systematically integrates three complementary mechanisms. Unlike prior GAN-based recommenders that rely exclusively on latent embeddings, GaSimGAN combines a MF generator that explicitly leverages user-item similarity constraints with an autoencoder-based discriminator, enabling richer representation learning. Genetic Algorithms (GA) serve dual roles: (i) a preprocessing phase that optimizes k-nearest neighbor selection via similarity-based search, performed once prior to training, and (ii) an offline one-shot hyperparameter optimization step [16] that identifies the optimal configuration $\theta = (\mu_G, \mu_D, \lambda_G, \lambda_D, m, \alpha)$ independently of the adversarial training pipeline. The resulting hyperparameters are then fixed throughout the entire GAN training phase, where only Adam-based gradient updates are applied. Finally, Pearson correlation constructs a robust similarity matrix that both stabilizes GAN training and enhances user-item modeling

The principal contributions of this work are threefold:

- We introduce GaSimGAN, a collaborative filtering framework that unifies GAN-based generation, GA optimization, and Pearson similarity modeling within a coherent adversarial architecture.
- We enhance the discriminator through autoencoder-based reconstruction, while equipping the generator with neighbor-informed MF to synthesize realistic and personalized user/item preferences.
- We propose a dual-phase GA strategy operating at two distinct stages: (i) a preprocessing phase for GA-based k-nearest neighbor selection via Pearson similarity, and (ii) an offline one-shot hyperparameter optimization step conducted independently of the adversarial training pipeline. While GA has been applied individually to similarity modeling and hyperparameter tuning in isolation, their systematic integration within a unified GAN-based collaborative filtering framework has not been previously explored.

The remainder of this paper is organized as follows. Section 2 discusses related works, while Section 3 presents our proposed collaborative filtering process and learning algorithm. In Section 4, we expose and analyze the experimental results, and finally, a conclusion closes the paper.

2. Related Works

Recommendation systems face persistent challenges including data sparsity, cold-start scenarios, and the need to capture complex nonlinear user-item relationships. While matrix factorization (MF), generative adversarial networks (GANs), and genetic algorithms (GA) have each addressed these issues independently, their integration remains largely unexplored. This section examines how prior work has addressed these challenges, revealing systematic limitations that motivate GaSimGAN's unified framework.

2.1. Modeling Complex user-item Interactions

The evolution of collaborative filtering has been driven by efforts to capture increasingly complex user-item relationships. Classical MF [17, 18] decomposes interaction matrices into latent features, offering computational efficiency and interpretability. However, its linear assumptions limit expressiveness in modeling nonlinear patterns.

Hybrid approaches have sought to address this limitation by integrating MF with deep learning. Song & Wang (2022) [19] combined MF with multilayer perceptrons to capture nonlinear interactions, while Sami et al. 2024 [20] fused collaborative filtering, neural CF, recurrent networks, and content-based filtering, achieving substantial improvements on MovieLens100k. Despite enhanced representational capacity, these models rely on static neighborhood definitions and lack adaptive optimization during training.

Generative adversarial networks have introduced a paradigm shift by framing recommendation as adversarial learning. GANs have demonstrated remarkable versatility across many application domains. From Image segmentation [21] to time series analysis [22] and fraud detection [23], GANs effectively address data scarcity through synthetic generation. In RS, this capability enables the synthesis of realistic user-item interactions and offers new strategies to mitigate sparsity and leverage implicit feedback.

IRGAN [12] pioneered adversarial learning for recommendation, treating it as a minimax game between generative and discriminative retrieval models. Building on this foundation, specialized architectures emerged. Chae et al. developed CFGAN [24], employing vector-wise adversarial training to reduce bias, and CAAE [25], which replaced MF generators with autoencoders while integrating Bayesian Personalized Ranking. More recently, Dervishaj et al. [26] proposed GANMF, embedding MF within autoencoder-based discriminators and demonstrating measurable improvements over classical MF. In this paper [27], authors introduced WGANMF-DO, a novel RS framework that integrates matrix factorization with a Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) to jointly optimize recommendation accuracy and diversity in implicit feedback settings. The proposed approach formulates the top-N recommendation task as a dual-objective optimization problem, addressing the limitations of traditional collaborative filtering models that primarily focus on accuracy and often yield redundant recommendation lists. The generator G is implemented as a matrix factorization model that learns low-dimensional latent representations for users and items resulting in continuous-valued user preference profiles suitable for ranking-based recommendation. The discriminator D is implemented as an autoencoder acting as a critic rather than a binary classifier. Given an input profile x, either sampled from the empirical data distribution p_{data} or generated by G, the autoencoder reconstructs it as $\hat{x} = D(x)$. The critic score is defined as the negative reconstruction error. This formulation provides a smooth and informative learning signal that improves adversarial training stability. To mitigate common adversarial issues such as mode collapse and vanishing gradients, the framework adopts the Wasserstein distance with gradient penalty. The generator is trained using a dual-objective loss that explicitly balances recommendation accuracy and diversity.

Comparative studies by Filali et al. [13, 28] revealed that GANs excel at capturing nonlinear relationships compared to variational autoencoders and diffusion models, but face training instability and scalability challenges. Persistent issues remain: hyperparameter sensitivity, unstable convergence, and underutilization of explicit similarity information. GaSimGAN directly addresses these limitations through its dual GA mechanism and Pearson-based similarity integration.

Quantum and evolutionary methods have explored alternative representation strategies. Banouar et al. [29] combined tensor nuclear norm minimization with quantum K-means clustering for sparsity handling, while Lara-Cabrera et al. [30] evolved entire MF architectures using genetic programming. Rezaei et al. [31] applied GA to enhance MF initialization. While these approaches showcase adaptability, they remain confined to preprocessing rather than adaptive training-time optimization. GaSimGAN synthesizes these directions by embedding MF within a GAN generator, leveraging GA for one-shot offline hyperparameter optimization [16] and preprocessing-based neighbor selection, and incorporating Pearson correlation to stabilize similarity-aware learning throughout training.

2.2. Addressing Data Sparsity and Cold-Start Challenges

Data sparsity and cold-start users pose fundamental obstacles in collaborative filtering. Traditional MF struggles under extreme sparsity due to insufficient interaction signals. Privacy-preserving extensions have broadened MF's applicability while maintaining robustness. Khan et al. [32] introduced fractional stochastic gradient descent (EFSGD) technique with elastic-net regularization for chaotic feedback environments, Wang et al. [33] developed piecewise mechanisms for local differential privacy, and Zhang et al. [34] proposed MENTOR, a mixture-model framework achieving superior RMSE (Root Mean Squared Error) and F-measure under privacy constraints. While

these methods enhance robustness in sensitive domains, they preserve MF's linear structure and do not explicitly address sparsity through data augmentation.

Generative approaches offer complementary solutions by synthesizing training data. Rating augmentation models such as RAGAN and RAGAN-BT [35] generate synthetic ratings to alleviate sparsity, while Wang et al.'s AugCF [36] generalizes augmentation through conditional GANs. These methods demonstrate GANs' potential for data synthesis, yet they operate independently of similarity modeling and adaptive optimization.

Scalability and cold-start handling have been addressed through parallelization and incremental learning. Yu et al. 2014 [37] developed CDD++, achieving significant speedups on billion-scale datasets, while Al-Sabaawi et al. 2021 [38] proposed Simultaneous Incremental MF (SIMF) for dynamic cold-start scenarios. Rendle et al. 2012 [39] introduced BPR-Opt, directly optimizing MF for ranking with implicit feedback. Banouar et al. 2018 [40] applied nuclear norm minimization for matrix completion in missing-data scenarios, and quantum-inspired approaches [8] have further reduced complexity in large-scale environments. Graph-based methods such as LightGCN [41] have shown strong performance on dense datasets, but their embedding propagation becomes unreliable under extreme sparsity conditions, limiting their effectiveness in sparse recommendation scenarios [42, 43]. However, these methods do not integrate generative augmentation or adaptive optimization strategies. GaSimGAN addresses sparsity through adversarial data synthesis while employing GA to adaptively select informative neighbors, enabling robust performance even under extreme sparsity conditions.

2.3. Optimization Strategies and Similarity Modeling

Effective recommendation requires balancing model capacity with training stability, a challenge compounded by the need for accurate similarity modeling. Traditional MF employs static optimization schemes (SGD, ALS) that do not adapt during training. Genetic algorithms have been applied primarily as preprocessing tools for similarity optimization. Kim & Ahn (2008) [44] used GA for user clustering, outperforming traditional k-means, while Bobadilla et al. 2011 [45] optimized similarity weights to improve accuracy in user-based CF. Xiao et al. 2015 [46] refined item-based similarity functions with GA, demonstrating significant prediction gains.

Subsequent work integrated additional modeling strategies. Salehi (2014) [47] optimized latent feature weights using GA to alleviate sparsity, Jia et al. 2014 [48] incorporated trust-aware weighting into GA-driven similarity models, and Lv et al. 2016 [49] extended GA to item-feature weighting in ontology-driven recommenders. Alhijawi & Kilani (2020) [50] proposed BLIGA, where GA evolves full recommendation lists rather than individual items, improving semantic alignment. Kilani et al. 2018 [51] hybridized GA with neighborhood-based CF to accelerate similarity computation. More recently, El-Ansari et al. 2023 [52] introduced mixed-variable GA training for neural networks, targeting accuracy, diversity, and coverage simultaneously.

Despite these advances, two critical limitations persist. First, GA is typically applied to a single objective, either similarity computation or hyperparameter tuning, but never systematically to both within the same framework. This prevents exploiting the complementary benefits of evolutionary optimization across multiple stages of the recommendation pipeline. Second, adversarial training in GAN-based recommenders operates with fixed similarity definitions and manually tuned hyperparameters, limiting both personalization and training stability. Sami et al. 2024 [20] incorporated adversarial components into hybrid recommenders but relied on static similarity structures. Deldjoo et al. 2025 [15] emphasized that generative recommenders introduce risks such as bias amplification and hallucination, advocating for holistic evaluation frameworks that balance effectiveness with ethical responsibility.

GaSimGAN addresses these limitations through three key innovations. First, it introduces a dual-phase GA mechanism: (i) selecting k-nearest neighbors during preprocessing based on Pearson correlation, performed once prior to training, and (ii) an offline one-shot hyperparameter optimization step [16] that identifies the optimal configuration $\theta = (\mu_G, \mu_D, \lambda_G, \lambda_D, m, \alpha)$ independently of the adversarial training pipeline. Unlike prior work that applies GA to a single objective, GaSimGAN systematically leverages GA across two distinct stages, ensuring both neighborhood structure and model initialization are jointly optimized. Second, it embeds Pearson correlation-based adaptive similarity matrices directly into the adversarial loss, guiding training and enhancing personalization. Third, it combines MF-based generation with autoencoder discrimination within a unified adversarial framework, enabling richer nonlinear representations while maintaining training stability. This synthesis extends prior approaches toward a more adaptive, stable, and similarity-aware recommendation system.

While prior work has made substantial progress in representation learning, sparsity handling, and optimization, three systemic gaps remain. First, MF models are predominantly linear with static similarity definitions, constraining their capacity to capture intricate user-item relationships. Second, GAN-based recommenders suffer from training instability, hyperparameter sensitivity, and underutilization of explicit similarity information. Third, GA is typically confined to a single optimization objective rather than integrated across multiple pipeline stages.

GaSimGAN directly addresses these limitations by: (i) embedding MF within a GAN framework to capture nonlinear user-item relationships while maintaining computational efficiency, (ii) incorporating Pearson correlation-based similarity constraints to stabilize adversarial training and enhance personalization, and (iii) employing GA in two distinct one-shot roles: a preprocessing step for neighbor selection performed once prior to training, and an offline hyperparameter optimization step that identifies the optimal configuration independently of the adversarial training pipeline. This unified framework synthesizes the strengths of MF, adversarial learning, and evolutionary optimization, extending prior approaches toward a more adaptive, robust, and similarity-aware collaborative filtering system.

3. Proposed approach: personalized Matrix factorization through GAN based architecture and Pearson similarity

3.1. Notation and problem formulation

Table 1. Summary of key notation

Notation	Description
$\mathcal{U}, \mathcal{I}$	Set of users and items
$\lvert U\rvert, \lvert I\rvert$	Number of users and items
$R \in \mathbb{R}^{\lvert U\rvert \times \lvert I\rvert}$	User–rating matrix (URM)
$X \in \{0,1\}^{\lvert U\rvert \times \lvert I\rvert}$	User–item interaction matrix (UIM)
$r_{u,i}$	Observed rating of user u for item i
$\hat{r}_{u,i}$	Predicted rating of user u for item i
$\bar{r}_u$	Average rating given by user u
$S \in \mathbb{R}^{\lvert U\rvert \times \lvert U\rvert}$	User–user similarity matrix (Pearson)
$S \in \mathbb{R}^{\lvert I\rvert \times \lvert I\rvert}$	Item–item similarity matrix (Pearson)
k	Number of nearest neighbors
U_y	Set of k users most similar to user y
I_i	Set of k items most similar to item i
f	Number of latent factors (embedding dimension)
$G(\cdot)$	Generator function
$D(\cdot)$	Discriminator function (reconstruction error)
L_G, L_D	Loss functions for generator and discriminator
μ_G, μ_D	Learning rates for generator and discriminator
λ_G, λ_D	Regularization coefficients
α	Feature matching coefficient
m	Margin coefficient (hinge loss)
θ	Hyperparameter configuration
$\mathcal{V}$	Validation set of users
N_p, N_g	GA population size and number of generations

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|U|}\}$ denote the set of $|U|$ users and $\mathcal{I} = \{i_1, i_2, \ldots, i_{|I|}\}$ the set of $|I|$ items. We define the User–Rating Matrix (URM) $R \in \mathbb{R}^{|U|\times|I|}$, where each entry $R_{u,i} = r_{u,i}$ represents the observed rating given by user u to item i. Missing ratings (unobserved user–item pairs) are denoted by $r_{u,i} = 0$.

To distinguish between observed and missing interactions, we define the User–Item Interaction Matrix (UIM) $X \in \{0,1\}^{|U|\times|I|}$ as a binary indicator:

$$X_{u,i} = \begin{cases} 1, & \text{if user } u \text{ has rated item } i, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Our objective is to predict missing entries $\hat{r}_{u,i}$ for all unobserved user–item pairs where $X_{u,i} = 0$, and subsequently generate personalized top-N recommendations for each user. We denote by $\hat{r}_{u,i}$ the predicted rating (or interaction score) that user u would assign to item i, where higher values indicate stronger preference. Our approach GaSimGAN operates in two variants: *GaSimGAN_user* for user-based collaborative filtering, which generates synthetic user profiles by leveraging similar users, and *GaSimGAN_item* for item-based collaborative filtering, which generates synthetic item profiles by leveraging similar items. The item-based variant is obtained by transposing the URM, i.e., $R^T \in \mathbb{R}^{|I|\times|U|}$, and applying the same framework with items as the primary entities.

3.2. Architecture Overview

GaSimGAN addresses the challenges of data sparsity, cold-start problems, and personalization in collaborative filtering through a novel framework that systematically integrates three complementary mechanisms. Unlike prior GAN-based recommenders that rely exclusively on latent embeddings, GaSimGAN combines similarity-driven matrix factorization with adversarial training and evolutionary optimization, enabling richer representation learning while maintaining training stability. Figure 1 illustrates the complete GaSimGAN workflow, which operates through three sequential phases:

- **Phase 1: Similarity Matrix Construction** (Section 3.3): construction of user–user (or item–item) similarity matrix using Pearson correlation.
- **Phase 2: Genetic Neighbor Selection** (Section 3.4): adaptive neighborhood optimization using a genetic algorithm.
- **Phase 3: GAN-based Profile Generation** (Section 3.5): adversarial synthesis of realistic user or item profiles to mitigate sparsity.

The GA serves a dual role in GaSimGAN: beyond neighbor selection in Phase 2, it identifies the optimal hyperparameter configuration $\theta = (\mu_G, \mu_D, \lambda_G, \lambda_D, m, \alpha)$ through an offline one-shot optimization step conducted prior to and independently of the adversarial training pipeline. This preliminary optimization ensures the model is initialized with dataset-specific hyperparameters, preventing mode collapse and enhancing convergence stability throughout training.

Training proceeds iteratively: the discriminator learns to reconstruct observed profiles with low error while assigning high reconstruction error to generated profiles; simultaneously, the generator improves its outputs to minimize the discriminator's reconstruction error on generated profiles. Once adversarial equilibrium is reached, when the discriminator can no longer reliably distinguish between observed and generated profiles, the generator produces realistic predictions suitable for top-N recommendation.

This three-phase architecture enables GaSimGAN to comprehensively adapt to diverse datasets: Phase 1 establishes similarity-based relationships, Phase 2 optimizes data representation through intelligent neighbor selection, and Phase 3 leverages adversarial training with continuous hyperparameter tuning to generate high-quality recommendations. The following sections detail each phase.

3.3. Phase 1: Pearson Similarity Computation

The framework begins by constructing a binary User–Item Interaction Matrix (UIM) from the User–Rating Matrix (URM) to identify which user–item pairs have been observed. Using the URM ratings and the UIM structure, the framework then computes a user–user (or item–item) similarity matrix S using Pearson correlation. This matrix captures preference relationships based on rating patterns while accounting for individual rating biases, serving as the foundation for neighborhood-based collaborative filtering in subsequent phases.

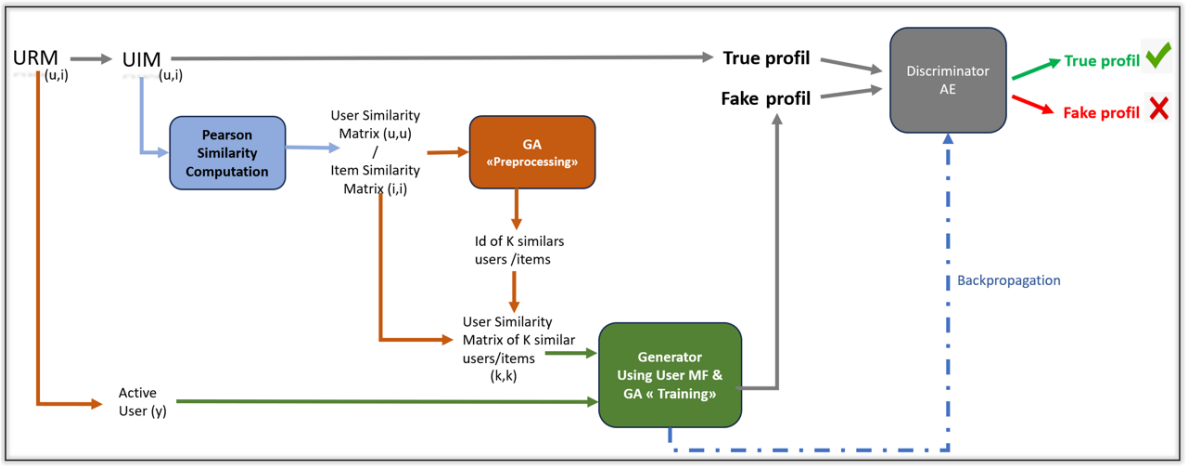

Figure 1. Architecture and workflow of GaSimGAN.

We compute user–user similarity using Pearson correlation [53], which centers ratings around each user's mean to account for individual rating biases. For two users u and v:

$$S_{u,v} = \frac{\sum_{i \in \mathcal{I}_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{u,v}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in \mathcal{I}_{u,v}} (r_{v,i} - \bar{r}_v)^2}} \tag{2}$$

where $\mathcal{I}_{u,v}$ denotes items co-rated by both users, and $\bar{r}_u$ is user u's average rating. The resulting similarity matrix $S \in \mathbb{R}^{|U| \times |U|}$ has values in $[-1, 1]$.

For item-based filtering, we compute $S \in \mathbb{R}^{|I| \times |I|}$ by applying the same formula with users and items swapped. This similarity matrix then guides the genetic algorithm's neighbor selection.

3.4. Phase 2: Genetic Algorithm for Identifying k Similar Users/Items

Rather than using a fixed neighborhood size k or manually tuning this parameter, GaSimGAN employs a genetic algorithm to optimize the selection of the k most similar users (or items) for each target entity. The GA evolves both the weighting scheme for similarity computation and the neighborhood size k by minimizing prediction error on the training set. This adaptive approach constructs localized neighborhood structures $U_y = \{u_1, u_2, \ldots, u_k\}$ tailored to each user y, where neighbors are selected based on their Pearson similarity $S_{y,j}$ from Phase 1.

This second phase employs a genetic algorithm to optimize the selection of the k most similar users (or items) for each target entity. Rather than using a fixed neighborhood size or manually tuning k, the GA evolves both the neighborhood size and the weighting scheme to minimize prediction error, constructing localized neighborhood structures tailored to each user.

3.4.1. Genetic representation

This approach introduces a divergence measure for users' rating habits, and a Genetic Algorithm (GA) is employed as an optimization tool to determine an appropriate weight vector for the similarity measure. In this technique, we adopt a real-number coding scheme. The weight vector $\mathbf{W} = (w_1, w_2, \ldots, w_{|V|})$ is represented as a real-valued vector of $|V|$ genes, where each gene corresponds to the importance of a particular combination in the vector V.

3.4.2. Genetic algorithm of the user/item similarity matrix

- **Initial Population:** Each population member represents a potential solution, encoded by one chromosome: k (number of user neighbors as in Eq 3) in the other methods.

To enhance solution diversity, randomness is introduced during the initial population generation.

- **Fitness Function:** The mean absolute error of the database (specifically the training set) is utilized as the fitness function, involving the following steps:
  1. **Similarity Computation:** Obtain the similarity of every pair of users.
  2. **Nearest Neighbors:** Find the nearest k neighbors of each user u, denoted as $N(u)$.
  3. **Prediction for Training Set Users:** For each item i in the training set, predict the ratings of user u over item i using the following formula:

$$\text{prediction}_{u,i} = \bar{r}_i + \frac{\sum_{n \in N(u)} \text{Similarity}(u,n) \cdot (r_{n,i} - \bar{r}_n)}{\sum_{n \in N(u)} |\text{Similarity}(u,n)|} \tag{3}$$

     Where:
     - $\text{prediction}_{u,i}$: the predicted rating for user u on item i.
     - $\bar{r}_u$: the average rating given by user u across all items.
     - $N(u)$: the set of k similar neighboring users to user u.
     - $\text{Similarity}(u,n)$: the similarity between user u and neighbor user n. This is computed using Pearson correlation.
     - $r_{n,i}$: the actual rating given by neighbor user n to item i.
     - $\bar{r}_n$: the average rating given by neighbor user n across all items.
  4. **Fitness Calculation:** The fitness function MAE (Mean Absolute Error) of the individual on the training set is computed as follows:

$$\begin{aligned} \text{fitness}(\text{Ind}) = 1 - \frac{1}{|U| \times |I|} \\ \times \sum_{u=1}^{|U|} \sum_{i=1}^{|I|} |\text{prediction}_{u,i} - r_{u,i}| \end{aligned} \tag{4}$$

     This fitness function measures the accuracy of the RS in predicting user ratings on the training set. The GA evolves both the weighting vector W and the neighborhood size k in order to minimize MAE, thereby improving the overall recommendation accuracy of the system. MAE was chosen as the fitness function because it provides a simple, continuous, and consistent objective, which facilitates convergence during the GA search.
- **Selection:** The efficiency of a genetic algorithm depends on the selection process, where individuals with higher fitness values have a greater chance of being chosen. This increases the likelihood of propagating favorable traits to future generations.
- **Crossover:** The one-point crossover, with a probability of 0.8, swaps genetic material between two parent chromosomes at a random point. This promotes the creation of offspring combining traits from both parents, enhancing genetic diversity and solution exploration.
- **Mutation:** The mutation operation introduces randomness by replacing one element of a chromosome with a new value when the mutation condition is met. With a probability of 0.01, it balances preserving genetic integrity while adding variability to avoid premature convergence.
- **Checking the Stop Condition:** The genetic algorithm runs until a termination condition is met, either when an individual reaches a predefined fitness threshold or when the maximum number of generations is reached.

3.5. Phase 3: GAN-based Similarity Matrix Factorization and GA

The core of GaSimGAN is a Generative Adversarial Network that synthesizes realistic user (or item) profiles. The generator G is conditioned on user y and leverages the neighborhood structures from Phase 2 to produce predicted interactions. This design stabilizes training while preserving the capacity to model complex nonlinear user-item relationships.

Algorithm 1 Genetic Algorithm for Neighbor Selection

Input: Similarity matrix S, rating matrix R, N_p, N_g
Output: Optimal k^* and neighborhoods N

```
Initialize population P with random k values
for gen = 1 to N_g do
    Evaluate fitness of each k ∈ P using k-NN predictions on R
    Select, crossover, and mutate to create offspring
    Replace population with offspring and elites
end for
k* ← best individual
Construct N: for each user, select top-k* neighbors from S
return k*, N
```

The discriminator D employs an autoencoder architecture to distinguish between observed profiles from the UIM and generated profiles from G. Unlike traditional binary classifiers, the autoencoder-based discriminator provides richer gradients by measuring reconstruction error, thereby facilitating more effective generator training.

In this framework, the GAN framework consists of two components: a generator G and a discriminator D. The generator is conditioned on user ID y to produce synthetic user profiles that align with y's characteristics, leveraging the Pearson similarity matrix computed during preprocessing. The discriminator is trained to distinguish between generated profiles and observed profiles from the User-item Interaction Matrix (UIM). Training proceeds iteratively, refining both components until the discriminator can no longer reliably differentiate between real and generated profiles. At the equilibrium, the generator produces realistic user profiles suitable for generating personalized recommendations.

3.5.1. Generator In the GaSimGAN_user framework, the generator G is a conditional generator incorporating a conditioning attribute y for each user. Diverging from conventional conditional GANs (cGANs), GaSimGAN excludes the noise vector z from the input, resulting in a deterministic mapping from the conditioning attribute to the generated user profile. This design stabilizes training while maintaining the capacity to model complex user–item relationships through similarity-weighted factorization.

The generator leverages the user similarity matrix and the user–item interaction matrix (UIM). Two embedding layers, $V \in \mathbb{R}^{|U|\times f}$ and $Z \in \mathbb{R}^{|U|\times f}$, represent latent factors for users, where f denotes the number of latent features. The training data comprise the k-nearest neighbor structures obtained through the genetic algorithm and the observed user–item interactions from the UIM.

GaSimGAN_user

The generator produces a vector $\hat{\mathbf{r}}_y$ representing predicted ratings for all items based on user y:

$$G(y) = \hat{\mathbf{r}}_y = \begin{bmatrix} \hat{r}_{y,1} \\ \vdots \\ \hat{r}_{y,i} \\ \vdots \\ \hat{r}_{y,|I|} \end{bmatrix} = \begin{bmatrix} \sum_{j\in U_y} r_{j,1} \cdot \text{SIM}_{y,j} \\ \vdots \\ \sum_{j\in U_y} r_{j,i} \cdot \text{SIM}_{y,j} \\ \vdots \\ \sum_{j\in U_y} r_{j,|I|} \cdot \text{SIM}_{y,j} \end{bmatrix} \tag{5}$$

where:

- $\hat{r}_{y,i}$ denotes the predicted interaction of active user y with item i
- $r_{j,i}$ denotes the observed interaction of user j (similar to y) with item i
- U_y represents the set of k users most similar to active user y
- $\text{SIM}_{y,j}$ represents the Pearson similarity between users y and j

GaSimGAN_item

For the item-based variant, the generator produces predicted interactions of all users with item i:

$$G(i) = \hat{\mathbf{r}}_i = \begin{bmatrix} \hat{r}_{1,i} \\ \vdots \\ \hat{r}_{u,i} \\ \vdots \\ \hat{r}_{|U|,i} \end{bmatrix} = \begin{bmatrix} \sum_{j \in I_i} r_{1,j} \cdot \text{SIM}_{i,j} \\ \vdots \\ \sum_{j \in I_i} r_{u,j} \cdot \text{SIM}_{i,j} \\ \vdots \\ \sum_{j \in I_i} r_{|U|,j} \cdot \text{SIM}_{i,j} \end{bmatrix} \tag{6}$$

where:

- $\hat{r}_{u,i}$ denotes the predicted interaction of user u with item i
- $r_{u,j}$ denotes the observed interaction of user u with item j (similar to i)
- I_i represents the set of k items most similar to item i
- $\text{SIM}_{i,j}$ represents the Pearson similarity between items i and j

This approach effectively utilizes the latent factors and similarity matrix to generate personalized predictions for user-item interactions within the GaSimGAN framework. Genetic Algorithms (GA) serve dual roles: first, selecting the k most similar users during preprocessing to construct neighborhood structures, performed once prior to training; second, identifying the optimal hyperparameter configuration $\theta = (\mu_G, \mu_D, \lambda_G, \lambda_D, m, \alpha)$ through an offline one-shot optimization step conducted independently of the adversarial training pipeline.

To deceive the discriminator, the generator minimizes the reconstruction error of the discriminator on generated profiles:

$$L_G(x, y) = D(G(y)) + \lambda_G ||\Omega_G||_2^2 \tag{7}$$

where $D(\cdot)$ and $G(\cdot)$ denote the discriminator and generator functions, respectively, x is the observed user profile corresponding to user y, λ_G is the L_2 regularization coefficient, and Ω_G is the set of parameters for the generator. The use of embedding layers limits the number of parameters to be learned by the generator to $\Omega_G = (V, Z)$.

To enhance training stability and address the limitation of producing a single deterministic profile for each user, we incorporate a feature matching loss term into the generator's loss function. Feature matching, associated with maximum mean discrepancy, represents the distance between two probability distributions [26]. The modified loss function is:

$$\begin{aligned} L_G(x, y) = (1 - \alpha) D(G(y)) \\ + \alpha \|\text{Enc}(x) - \text{Enc}(G(y))\|_2^2 + \lambda_G \|\Omega_G\|_2^2 \end{aligned} \tag{8}$$

Here, $\text{Enc}(\cdot)$ denotes the encoder output in the discriminator's autoencoder, x is the observed user profile corresponding to y, and $\alpha \in [0, 1]$ is a constant balancing adversarial and feature matching losses. A bottleneck autoencoder is employed in the discriminator, enabling it to learn salient features within its encoding layer. Minimizing L_G ensures that generated profiles conform to the distribution of observed profiles in the latent space induced by the encoding layer. Note that $\text{Enc}(\cdot)$ is not frozen; its parameters are updated during training as part of the discriminator via L_D, allowing the feature matching loss to adapt as the discriminator's representation evolves.

3.5.2. Hyperparameter optimization via GA Beyond neighbor selection, the GA optimizes the hyperparameter set $\theta = (\mu_G, \mu_D, \lambda_G, \lambda_D, m, \alpha)$ (Table 5) through an offline one-shot optimization step conducted prior to and independently of the adversarial training pipeline. The optimal θ identified by the GA is fixed throughout the entire training phase, during which the Adam optimizer updates the generator and discriminator weights exclusively through gradient descent. This preliminary optimization ensures that gradient-based updates operate from an optimal starting configuration, preventing mode collapse and enhancing convergence stability across all datasets.

Each configuration θ is evaluated on the validation set using:

$$\text{fitness}(\theta) = 1 - \text{MAE}_V(\theta) \tag{9}$$

where $\text{MAE}_V(\theta)$ denotes the mean absolute error on the validation set.

3.5.3. Discriminator

In the GaSimGAN_user Discriminator, a departure from the traditional binary classifier discriminator of conditional Generative Adversarial Networks (cGAN) is introduced. Instead, an autoencoder is employed to provide more nuanced gradients for the generator. This autoencoder receives a user profile as input, which could originate from the user item interaction matrix or be generated by the GaSimGAN generator. The autoencoder's output is the reconstructed version of the input profile. This approach draws inspiration from EBGAN (Energy-Based GAN), which initially introduced an autoencoder-based discriminator as an energy function [26]. In EBGAN, the energy function assigns low energy to data from the training set and high energy to data generated by the generator, thereby aiding in distinguishing the input source for the discriminator. In the context of GaSimGAN, the reconstruction error of the autoencoder serves as the energy function in a similar manner, facilitating the discrimination process. In the GaSimGAN framework, the energy function for the discriminator is determined by the reconstruction error of the autoencoder:

$$D(x) = ||\mathcal{D}_{\text{dec}}(E_{\text{enc}}(x)) - x||^2 \tag{10}$$

Here, $\mathcal{D}_{\text{dec}}(\cdot)$ and $E_{\text{enc}}(\cdot)$ represent the decoder and encoder functions of the autoencoder, respectively, and $||\cdot||^2$ denotes the Euclidean norm.

For a given user identifier y and its authentic profile x, the discriminator D minimizes a hinge loss function:

$$\mathcal{L}(x, y) = \mathcal{D}(x) + [mD(x) - D(G(y))]^+ + \lambda_D||\Omega_D|| \tag{11}$$

In this equation:

- $[\cdot]^+ = \max(0, \cdot)$,
- $D(\cdot)$ is the discriminator function,
- $G(\cdot)$ is the generator function,
- λ_D is a regularization coefficient,
- Ω_D represents the set of parameters for D.

Differing from EBGAN, GaSimGAN employs a positive margin coefficient m instead of a positive margin. This choice is made to constrain the range of its values.

3.5.4. Algorithm proposed for GaSimGAN Training

Algorithm 2 presents the complete training procedure for GaSimGAN_user (the item-based variant follows the same procedure on R^T).

4. Experiments

4.1. Experimental process

We have used GaSimGAN to evaluate recommendation systems on three datasets with different characteristics, shown in Table 2. ML-1M [54] provides explicit user ratings for movies, making it widely used in collaborative filtering research. MovieLens HetRec [55] extends this with IMDb and Rotten Tomatoes metadata, enabling content-based and hybrid recommendations. LastFM [55] captures implicit user preferences through listening counts, making it ideal for studying music recommendations. These datasets offer diverse evaluation scenarios, allowing GaSimGAN to demonstrate its effectiveness across different recommendation settings.

Table 2 summarizes the main characteristics of the datasets used in our experiments. It reports the number of interactions, users, items, feedback type, as well as the density and sparsity levels. These statistics highlight the differences in scale and sparsity across datasets, which directly influence model performance and convergence behavior.

Algorithm 2 GaSimGAN_user Training Procedure

Input: $R \in \mathbb{R}^{|U|\times|I|}$, $X \in \{0,1\}^{|U|\times|I|}$, $\mathcal{V}$, $\tau = 5$, GA settings, $E_{\max}$, B
Output: Trained G and D

// Phase 1: User–User Similarity (Section 3.3)
Compute $S \in \mathbb{R}^{|U|\times|U|}$ using Pearson ($\tau = 5$)
for each pair (u, v) with $|\mathcal{I}_{u,v}| \geq \tau$ **do**
    $S_{u,v} \leftarrow \text{PearsonCorr}(R[u,:], R[v,:])$
end for

// Phase 2: GA Neighbor Selection (Section 3.4)
$(k^*, N) \leftarrow \text{GA}(S, R, N_p, N_g)$
// Offline HPO: GA Hyperparameter Optimization
$\theta \leftarrow \text{GA\_HPO}(\mu_G, \mu_D, f, \lambda_G, \lambda_D, m, \alpha)$
// One-shot offline step, conducted independently of training
// Phase 3: GAN Training (Section 3.5)
Initialize G, D with fixed θ
for epoch = 1 to $E_{\max}$ **do**
    for each batch $\mathcal{B}$ of B users **do**
        $U_y \leftarrow N[y]\ \forall y \in \mathcal{B}$
        $\hat{r}_y \leftarrow G(y, U_y, S)\ \forall y \in \mathcal{B}$
        $r_y \leftarrow R[y,:]\ \forall y \in \mathcal{B}$
        Update D: $\Omega_D \leftarrow \Omega_D - \mu_D \nabla L_D$
        Update G: $\Omega_G \leftarrow \Omega_G - \mu_G \nabla L_G$
    end for
end for
return G, D =0

Table 2. Dataset statistics and features

Dataset	# Interactions	# Users	# Items	Feedback Type	Density (%)	Sparsity (%)
ML-1M	1,000,209	6,040	3,952	1–5 Ratings	4.25	95.75
HetRec2011	855,598	2,113	10,197	1–5 Ratings	4.00	96.00
LastFM	92,834	1,892	17,632	Listening Counts	0.28	99.72

We focus on implicit feedback scenarios. Movie ratings in MovieLens and listening counts in LastFM are binarized: ratings > 0 or counts > 0 are mapped to 1 (observed interaction), while unobserved entries remain 0, forming the binary interaction matrix. No filtering is applied to users or items in order to preserve the original dataset distributions. Each dataset is randomly divided into training and test sets using an $80/20$ split. To ensure meaningful evaluation, test users are required to have interacted with at least one item in the training set. Since the datasets contain no explicit negative feedback, all unobserved user–item pairs are treated as implicit negatives, and ranking is performed over the full candidate item set during evaluation.

4.2. Experimental setup

The experiments in this study were performed on a machine with the following specifications: Processor (CPU): 11th Gen Intel® Core™ i7-11800H @ 2.30GHz, 8 cores, 2.30 GHz max clock speed, Memory (RAM): 16 GB (2 x 8 GB), 3200 MHz, and Graphics Processing Unit (GPU): Intel® UHD Graphics (integrated) and NVIDIA GeForce RTX 3050 Ti Laptop GPU.
To ensure reproducibility, all experiments were conducted with a fixed random seed (1337). Due to the significant computational cost, multiple independent runs were not feasible.

4.3. Results and discussion

In the preprocessing stage, we evaluated the following methods to compute the similarity matrix: Cosine, Pearson, Jaccard, and K-means (with $k = 6$, determined by the Elbow Method). The similarity matrix is computed from the User-Item matrix. The results, shown in Table 3, indicate that Pearson achieved the best performance (bold values). Following standard practice in collaborative filtering [56, 57], Pearson correlation is computed only for entity pairs (user–user or item–item) sharing at least $\tau = 5$ common ratings, ensuring statistical reliability. Pairs with fewer co-rated items yield unstable similarity scores due to insufficient sample size. For hyperparameter optimization, we compared three metaheuristics [16]: Genetic Algorithm (GA), Simulated Annealing (SA), and Particle Swarm Optimization (PSO). The search space is summarized in Table 5. This offline one-shot optimization was conducted independently of the adversarial training pipeline, prior to the final model training.

As shown in Table 4, GA consistently achieves the highest MAP values on all datasets, and dominates all metrics on LastFM and ML HetRec. On ML-1M, PSO marginally outperforms GA on NDCG while GA leads on MAP. Based on these results, GA was selected as the HPO method for GaSimGAN, as MAP is the primary optimization objective in top-N recommendation.

The optimal hyperparameter configuration $\theta = (\mu_G, \mu_D, \lambda_G, \lambda_D, m, \alpha)$ identified by GA (summarized in Table 7) was fixed and used throughout the entire GAN training phase, during which only the Adam optimizer updates the generator and discriminator weights via gradient descent.

Table 3. Obtained results for similarity matrix

Method	**Time in sec**	**MAE**	**MSE**	**RMSE**
Cosine	18.031021	0.561273	0.559921	0.748279
Pearson	**15.601877**	**0.094140**	**0.068768**	**0.262236**
Jaccard	73.245884	0.175460	0.159457	0.399321
K_means	88.354207	0.163781	0.158080	0.397592

Table 4. Results of Metaheuristic Optimization Techniques Using GAN Across Three Datasets

Dataset	Method	MAP@5	NDCG@5	MAP@10	NDCG@10
1M ML	GA	**0.3084**	0.4079	**0.2445**	0.3740
	SA	0.2586	0.3618	0.2041	0.3356
	PSO	0.3048	**0.4113**	0.2438	**0.3808**
LastFM	GA	**0.1630**	**0.2452**	**0.1148**	**0.2132**
	SA	0.1538	0.2333	0.1077	0.2029
	PSO	0.1509	0.2296	0.1053	0.1977
ML HetRec	GA	**0.4819**	**0.5757**	**0.4030**	**0.5316**
	SA	0.2586	0.3618	0.2041	0.3356
	PSO	0.4590	0.5473	0.3876	0.5092

Table 6 presents the parameter settings of the Genetic Algorithm.

Table 7 summarizes the key hyperparameters obtained after optimization, including epochs, batch size, margin coefficient, learning rates, embedding dimension (Emb Dim) and the number of similar users k, which varied between 15 and 21. These settings were adapted to each dataset's scale and sparsity and were consistently selected during the hyperparameter search.

We assess our proposed model in comparison to four baseline models designed for addressing the top-N recommendation problem, which includes robust baselines [58, 59]

- CAAE [25]: As a GAN-based RS, CAAE incorporates BPR loss in the discriminator and employs two autoencoder-based generators. For CAAE, hyperparameter tuning is conducted based on the ranges provided by the authors.

Table 5. Hyperparameter Search Space

Hyper-parameter	**Type**	**Search space**
Number of epochs	Discrete	[1 , 250]
Number of latent factors (f in 3.5.1)	Discrete	[1 , 200]
Units in the coding layer of AE (3.5.3)	Discrete	[1 , 512]
Margin coefficient m (Eq 11)	Discrete	[1 , 10]
Batch size	Categorical value	{16, 32, 64, 128, 256, 512, 1024}
Feature matching coefficient α (Eq 8)	Continuous	[0.0001, 0.01]
Regularization coefficients λ_D (Eq 11) and λ_G (Eq 8)	Continuous	$[10^{-6}, 10^{-4}]$
Learning rates μ_D and μ_G	Continuous	[0.001, 0.1]

Table 6. GA parameter setting.

Parameter	Rate of selection	Rate of crossover	Rate of mutation	N_g	N_p
Setting	60%	80%	10%	10	10

Table 7. Key hyperparameter configurations per dataset after optimization

Dataset	**Epochs**	**Batch Size**	**Margin** m	(μ_D, μ_G)	**Emb Dim**
ML-1M	100	500	10	(0.0001, 0.00285)	500
HetRec2011	250	512	10	(0.0001, 0.00285)	1000
LastFM	300	500	4	(0.00018, 0.0014)	350

- CFGAN [24]: This GAN-based recommendation System (RS) proposes a vector-wise training for GAN in RS. We fine-tune all the hyperparameters within the ranges provided by the authors.
- WGAN-DO[27]: a Wasserstein Generative Adversarial Network with Gradient Penalty that incorporates a dual-objective optimization strategy to jointly balance accuracy and diversity during adversarial training. We fine-tune all the hyperparameters within the ranges provided by the authors.
- GANMF [26]: An approach utilizing Generative GANs to acquire latent factors associated with users and items within a MF framework, addressing the generic top-N recommendation challenge.

.

The experimental results and comprehensive comparison across three datasets (ML-1M, HetRec2011, LastFM) for various recommendation models and evaluation metrics are presented below. The models are evaluated based on MAP, Precision, and NDCG for cutoffs at 5 (see Table 8), at 10 (see Table 9), at 20 (see Table 10), and at 50 (see Table 11).

Table 8. Performance Metrics @ Cutoff 5 for Different Models Across Datasets

1***Models**	**ML-1M Dataset**			**HetRec2011 Dataset**			**LastFM Dataset**		
	PREC	**MAP**	**NDCG**	**PREC**	**MAP**	**NDCG**	**PREC**	**MAP**	**NDCG**
CAAE	0.2091	0.1531	0.2217	0.4574	0.3894	0.4767	0.0691	0.0504	0.0834
CFGAN_user	0.3817	0.3066	0.4044	0.4933	0.4151	0.5123	0.2127	0.1482	0.2358
GANMF_user	0.4161	0.3420	0.4405	0.5870	0.5255	0.6076	**0.2579**	0.2004	**0.2918**
GaSimGAN_user	**0.4356**	**0.3643**	**0.4537**	**0.6046**	**0.5330**	**0.6165**	0.2476	**0.2776**	0.2436
WGAN-DO_user	.4212	0.3498	0.4391	0.5893	0.5184	0.6012	0.2381	0.2689	0.2350
CFGAN_item	0.2122	0.1546	0.2211	0.4269	0.3533	0.4462	0.2006	0.1433	0.2219
GANMF_item	0.4226	0.3475	0.4460	0.6028	0.5445	0.6230	**0.2607**	0.1983	**0.2917**
GaSimGAN_item	**0.4322**	0.**3495**	**0.4504**	**0.6636**	**0.5539**	**0.6596**	0.2503	**0.2006**	0.2823
WGAN-DO_item	0.4187	0.3362	0.4360	0.6481	0.5394	0.6442	0.2410	0.1918	0.2725

Table 9. Performance Metrics @ Cutoff 10 for Different Datasets

2***Models/Metrics**	**ML-1M Dataset**			**HetRec2011 Dataset**			**LastFM Dataset**		
	PREC	**MAP**	**NDCG**	**PREC**	**MAP**	**NDCG**	**PREC**	**MAP**	**NDCG**
CAAE	0,1828	0.1170	0.2045	0.4038	0.3141	0.4347	0.0610	0.0354	0.0786
CFGAN_user	0.3252	0.2424	0.3697	0.4382	0.3401	0.4698	0.1608	0.1020	0.2049
GANMF_user	0.3576	0.2757	0.4056	0.5305	0.4500	0.5641	**0.1900**	0.1402	0.2513
GaSimGAN_user	**0.3692**	**0.2835**	**0.4127**	**0.5412**	**0.4518**	**0.6271**	0.1824	**0.1522**	**0.2505**
WGAN-DO_user	0.3558	0.2719	0.3996	0.5260	0.4382	0.6114	0.1741	0.1456	0.2420
CFGAN_item	0.1877	0.1191	0.2039	0.3769	0.2830	0.4062	0.1493	0.0990	0.1918
GANMF_item	0.3665	0.2830	0.4138	0.5437	0.4669	0.5784	**0.1889**	0.1388	**0.2495**
GaSimGAN_item	**0.3962**	**0.3265**	**0.4294**	**0.5447**	**0.4732**	**0.5805**	0.1799	**0.1454**	0.2387
WGAN-DO_item	0.3829	0.3148	0.4162	0.5296	0.4597	0.5651	0.1718	0.1389	0.2304

Table 10. Performance Metrics @ Cutoff 20 for Different Datasets

2***Models/Metrics**	**1M Dataset**			**HetRec2011 Dataset**			**LastFM Dataset**		
	PREC	**MAP**	**NDCG**	**PREC**	**MAP**	**NDCG**	**PREC**	**MAP**	**NDCG**
CAAE	0.1552	0.0913	0.1941	0.3453	0.2467	0.3902	0.0493	0.0378	0.0940
CFGAN_user	0.2654	0.1977	0.3487	0.3736	0.2695	0.4224	0.1199	0.1079	0.2338
GANMF_user	0.2964	0.2296	0.3867	0.4616	**0.3715**	0.5151	**0.1352**	0.1452	**0.2792**
GaSimGAN_user	**0.3002**	**0.2306**	**0.3880**	**0.4629**	0.3614	**0.5194**	0.1210	**0.2005**	0.1243
WGAN-DO_user	0.2884	0.2197	0.3751	0.4478	0.3492	0.5041	0.1148	0.1929	0.1186
CFGAN_item	0.1600	0.0928	0.1909	0.3313	0.2267	0.3707	0.1069	0.1021	0.2145
GANMF_item	0.3024	**0.2367**	**0.3946**	0.4715	**0.3866**	**0.5276**	**0.1339**	0.1439	**0.2770**
GaSimGAN_item	**0.3526**	0.2316	0.3963	**0.4724**	0.3506	0.5106	0.1256	**0.1503**	0.2604
WGAN-DO_item	0.3394	0.2209	0.3831	0.4573	0.3382	0.4951	0.1193	0.1437	0.2520

Table 11. Performance Metrics @ Cutoff 50 for Different Datasets

Models/Metrics	**1M**			**HetRec2011**			**LastFM**		
	PREC	**MAP**	**NDCG**	**PREC**	**MAP**	**NDCG**	**PREC**	**MAP**	**NDCG**
CAAE	0.1142	0.0742	0.1990	0.2627	0.1770	0.3426	0.0339	0.0433	0.1242
CFGAN_user	0.1898	0.1723	0.3582	0.2797	0.1936	0.3683	0.0739	0.1244	0.2899
GANMF_user	0.2127	**0.2036**	**0.4005**	**0.3522**	**0.2787**	**0.4564**	**0.0782**	**0.1619**	**0.3313**
GaSimGAN_user	**0.2130**	0.2035	0.4001	0.3199	0.2453	0.4554	0.0432	0.1465	0.3295
WGAN-DO_user	0.2021	0.1940	0.3878	0.3084	0.2350	0.4402	0.0398	0.1396	0.3201
CFGAN_item	0.1236	0.0764	0.1969	0.2558	0.1648	0.3295	0.0642	0.1149	0.2607
GANMF_item	0.2181	**0.2118**	**0.4105**	**0.3610**	**0.2927**	**0.4693**	**0.0759**	**0.1592**	**0.3253**
GaSimGAN_item	**0.3063**	0.1239	0.4096	0.3245	0.2034	0.4436	0.0533	0.1496	0.3213
WGAN-DO_item	0.2947	0.1178	0.3969	0.3121	0.1942	0.4289	0.0496	0.1429	0.3120

At first glance, the LastFM dataset tends to yield lower evaluation metrics compared to others. This is mainly because it relies on implicit feedback, such as listening data, rather than explicit ratings, making it harder to interpret user preferences. Additionally, the dataset's characteristics, including high sparsity, social dependencies, and the dynamic nature of music tastes, add to the challenge of generating accurate recommendations.The long-tail distribution, where a small number of songs receive the majority of interactions, further complicates model performance, as it limits the ability to generalize effectively. Given these complexities, models like GaSimGAN_item, which are designed to handle sparse and intricate data, perform better on this dataset. In contrast, the HetRec2011 dataset generally produces higher evaluation metrics compared to both the ML-1M and LastFM datasets. Moreover, as the cutoff value increases (e.g., from 5 to 50), precision tends to decline, while metrics such as MAP (Mean Average Precision) and NDCG often improve or remain stable. This trend is expected since

precision is highly sensitive to the number of recommendations, whereas MAP and NDCG focus more on ranking quality, making them more resilient to changes in the cutoff value.

The analysis reveals also that GaSimGAN_item generally outperforms GaSimGAN_user, particularly in more complex datasets like HetRec2011 and LastFM. These datasets are characterized by high sparsity, long-tail distributions, and diverse user preferences, making item-based modeling more effective. GaSimGAN_item demonstrates superior Precision, MAP, and NDCG, especially at higher cutoffs. However, in the ML-1M dataset, GaSimGAN_user performs competitively, particularly at lower cutoffs. This suggests that in denser datasets with more homogeneous user-item interactions, user-based modeling can be more effective in capturing immediate preferences. At higher cutoffs, the gap between the two models narrows, with GaSimGAN_item catching up or even surpassing GaSimGAN_user in some cases. These findings indicate that GaSimGAN_item is better suited for complex, sparse datasets with intricate item relationships, particularly when recommendations are made at larger cutoffs. Meanwhile, GaSimGAN_user is more effective in denser datasets, especially for capturing short-range user preferences at lower cutoffs. The choice between the two models should be guided by dataset complexity and the desired recommendation depth.

The results show that both the GaSimGAN_user and GaSimGAN_item models consistently outperform other models across different datasets, particularly in terms of precision and NDCG. For example:

- At Cutoff 5, GaSimGAN_user improves precision by 4.7% over GANMF_user in the ML-1M Dataset and by 3% in the HetRec2011 Dataset, but it shows a slight decline of -4% in the LastFM Dataset.
- Similarly, GaSimGAN_item improves performance by 2.3% in the ML-1M Dataset and by 10% in the HetRec2011 Dataset, while also showing a slight decline of -4% in the LastFM Dataset.
- The biggest improvement is seen in the HetRec2011 Dataset at Cutoff 5, where GaSimGAN_item achieves a 10% gain in precision over GANMF_item.

As the cutoff increases, the trends vary:

- At Cutoff 10, GaSimGAN_user outperforms GANMF_user by 3% in the ML-1M Dataset and by 2% in the HetRec2011 Dataset, but it slightly underperforms in Precision on the LastFM Dataset.
- The most significant improvement happens at Cutoff 20 in the ML-1M Dataset, where GaSimGAN_item shows a 17% improvement over GANMF_item.
- However, at Cutoff 50, there are some declines, especially in the HetRec2011 and LastFM Datasets. Notably, GANMF_item outperforms GaSimGAN_item at Cutoff 50 in both datasets in terms of MAP and NDCG, while GaSimGAN_item shows a strong 40% gain in Precision for the ML-1M Dataset, indicating better performance in large-scale movie recommendation scenarios. This underperformance at higher cutoffs can be attributed to the GA-optimized local neighborhoods, which are highly effective at lower cutoffs but may overfit to dense interaction regions, limiting generalization to larger recommendation lists on sparse datasets such as HetRec2011 and LastFM.

4.4. Computational Complexity and Scalability

The computational cost of GaSimGAN is primarily dominated by the adversarial training loop of the generator–discriminator architecture, with additional overhead introduced by the GA. The GA plays a dual role: it optimizes the selection of k-nearest neighbors during preprocessing (Phase 2) and identifies optimal hyperparameters through an offline optimization step conducted independently of the main training pipeline. The pipeline consists of three sequential phases. Phase 1 (Pearson similarity computation) is negligible, requiring only 15 seconds (ML-1M), 11 seconds (LastFM), and 25 seconds (ML HetRec), representing less than 0.07% of total training time. Phase 2 (GA-based k-nearest neighbor selection) is performed only once prior to training and added approximately 7.5 minutes (ML-1M), 2 minutes (LastFM), and 4.2 minutes (ML HetRec), corresponding to at most 2.08% of total training time. Phase 3 (final GAN training) dominates the runtime, requiring approximately 364 minutes (ML-1M, 100 epochs), 527 minutes (LastFM, 300 epochs), and 1,539 minutes (ML HetRec, 250 epochs), representing more than 97.9% of the total training time. The longer training time observed on LastFM (527 minutes) despite fewer users (1,884) is explained by its significantly larger item space (17,626 items), which increases the autoencoder

discriminator complexity $O(Bf)$ where f corresponds to the number of items. Similarly, ML HetRec requires 1,539 minutes due to the combination of a large item space (10,109 items) and a higher number of training epochs (250). Hyperparameter optimization using GA was conducted as a separate offline step, requiring approximately 78 minutes (ML-1M), 117 minutes (LastFM), and 103 minutes (ML HetRec), and is therefore not included in the reported training time. Table 12 provides a detailed empirical breakdown of the wall-clock training time across all datasets, confirming that the theoretical complexity analysis is consistent with the observed runtimes.

Table 12. Computational cost breakdown of GaSimGAN_user training pipeline across datasets

Phase	**ML-1M**	**LastFM**	**ML HetRec**
Phase 1 — Pearson Similarity	15 sec	∼11 sec	25 sec
Phase 2 — GA Preprocessing	∼7.5 min	∼2 min	∼4.2 min
Phase 3 — GAN Training	∼364 min	∼527 min	∼1,539 min
Total (Phase 1+2+3)	∼371.75 min	∼529.18 min	∼1,543.62 min
% Phase 1 of total	0.07%	0.03%	0.03%
% Phase 2 of total	2.02%	0.38%	0.27%
% Phase 1+2 of total	**2.08%**	**0.41%**	**0.30%**
% Phase 3 of total	97.92%	99.59%	99.70%
HPO GA (offline, not included)	∼78 min	∼117 min	∼103 min

The overall complexity of GaSimGAN is $O(T|U|(k+f)+N_pN_g)$, where T is the number of epochs, $|U|$ the number of users, k the latent dimension, f the feature size, and N_p, N_g the GA population size and number of generations. The dominant term $O(T|U|(k+f))$ corresponds to the adversarial training loop, where the generator performs matrix multiplication of complexity $O(Bk)$ and the discriminator (autoencoder) involves encoding and decoding of complexity $O(Bf)$ per batch. The GA term $O(N_pN_g)$ is fixed and independent of T, confirming that the preprocessing overhead remains less than 2.1% of total training time across all datasets. Phases 1 (Pearson similarity, $O(|U|^2 \cdot |I|)$), 2 (GA neighbor selection, $O(N_pN_g \cdot |U|)$), and HPO GA ($O(N_pN_g \cdot |U| \cdot |I|)$) are all conducted as independent one-shot offline steps prior to training and are therefore not included in the dominant complexity term.

5. Conclusion

In conclusion, we introduced a novel architectural approach, GaSimGAN, which integrates Generative Adversarial Networks (GANs) with model-based collaborative filtering for both user-based and item-based recommendation systems. Our method leverages the Genetic Algorithm (GA) alongside a personalized similarity matrix, which serves as input and refines the data used to generate top-N recommendations as output. By focusing on similar users or items, the personalized similarity matrix reduces noise, handles data sparsity, and enhances recommendation accuracy, ensuring more relevant and tailored results. The process unfolds in three main stages: first, the Pearson method is used to calculate similarity; then, GA identifies the k-nearest similar users; and finally, MF is applied in the GAN's generator to capture user–item relationships. Additionally, GA is employed a second time as an offline one-shot hyperparameter optimization step, identifying the optimal configuration $\theta = (\mu_G, \mu_D, \lambda_G, \lambda_D, m, \alpha)$ prior to and independently of the adversarial training pipeline. The resulting hyperparameters are fixed throughout the entire GAN training phase, during which only Adam-based gradient updates are applied. This hybrid approach improves recommendation accuracy by generating more realistic user profiles and enhancing overall system performance.

Our approach introduces two variants: *GaSimGAN_user* and *GaSimGAN_item*. The item-based variant transposes the User–Item Rating Matrix (URM) to focus on item similarity, which leads to significant improvements in precision (PREC), as demonstrated by a 40% improvement at cutoff 50 for the ML-1M dataset. In contrast, the user-based variant excels in ranking quality, achieving a 38% improvement in MAP@5 for the LastFM dataset,

despite a slight decline in Precision@5 (-4% vs. GANMF_user), reflecting a trade-off between ranking quality and precision on this sparse dataset.

By leveraging historical user interactions, GaSimGAN effectively learns user behavior patterns, enhancing recommendation accuracy. The integration of GANs, Genetic Algorithms, and similarity-personalized MF positions GaSimGAN as a promising approach in collaborative filtering. However, its performance varies across datasets, with strong results in precision-focused metrics (e.g., PREC) but challenges in ranking quality (e.g., NDCG) for sparser datasets such as LastFM. Future work will investigate statistical significance testing across multiple runs and cold-start performance for users and items with fewer than $\tau = 5$ co-rated interactions, as well as comparisons with recent non-GAN baselines such as graph-based models. Looking ahead, exploring a GaSimGAN-based quantum computing approach [60] could be an exciting avenue for optimizing processing and further boosting performance.

REFERENCES

1. B. Wang, and X. Zhang, *Personalized recommendation systems in the era of big data*, Procedia Computer Science, vol. 262, pp. 1115–1122, 2025.
2. Z. Xia, A. Sun, J. Xu, Y. Peng, R. Ma, and M. Cheng, *Contemporary recommendation systems on big data and their applications: A survey*, IEEE Access, vol. 12, pp. 196914–196928, 2024.
3. S. Raza, M. Rahman, S. Kamawal, A. Toroghi, A. Raval, F. Navah, and A. Kazemeini, *A comprehensive review of recommender systems: Transitioning from theory to practice*, Engineering Applications of Artificial Intelligence, vol. 124, pp. 106569, 2023.
4. M. Bakhshizadeh, C. Jilek, H. Maus, and A. Dengel, *Towards context-aware recommender systems for supporting knowledge workers in personal and corporate information space*, In Proceedings of INFORMATIK 2024, pp. 1323–1332, 2024.
5. Y. Li, J. Tang, M. S. Gönül, et al., *A novel approach for capturing consumer behaviour and preference: MulVAEK*, IEEE Computational Intelligence Magazine, 2025.
6. G. Zeng, J. Lu, and G. Zhang, *Are we really making recommendations robust? Revisiting model evaluation for denoising recommendation*, IEEE Computational Intelligence Magazine, 2025.
7. S. Filali-Zegzouti, O. Banouar, and M. Benslimane, *Recommendation systems techniques based on generative models and matrix factorization: a survey*, Mathematical Modeling and Computing, vol. 11, no. 4, 2024.
8. O. Ouedrhiri, O. Banouar, S. E. Hadaj, and S. Raghay, *Intelligent recommender system based on quantum clustering and matrix completion*, Concurrency and Computation: Practice and Experience, vol. 34, 2022.
9. O. Banouar, and S. Raghay, *Enriching SPARQL queries by user preferences for results adaptation*, International Journal of Software Engineering and Knowledge Engineering, vol. 28, no. 08, pp. 1195–1221, 2018.
10. Y. Wu, C. DuBois, A. X. Zheng, et al., *Collaborative denoising auto-encoders for top-N recommender systems*, In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp. 153–162, 2016.
11. D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, *Variational autoencoders for collaborative filtering*, In Proceedings of the 2018 World Wide Web Conference, pp. 689–698, 2018.
12. J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, *IRGAN: A minimax game for unifying generative and discriminative information retrieval models*, In Proceedings of the 40th International ACM SIGIR Conference, pp. 515–524, 2017.
13. S. Filali-Zegzouti, O. Banouar, and M. Benslimane, *Comparative analysis of deep learning-based generative models used for recommendation systems*, In Intelligent Systems and Advanced Computing Sciences, Springer Nature Switzerland, 2025.
14. L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu, H. Xiong, and E. Chen, *A survey on large language models for recommendation*, World Wide Web, vol. 27, no. 5, pp. 1781–1823, 2024.
15. Y. Deldjoo, N. Mehta, M. Sathiamoorthy, S. Zhang, P. Castells, and J. McAuley, *Toward holistic evaluation of recommender systems powered by generative models*, In Proceedings of the 48th International ACM SIGIR Conference, pp. 3932–3942, 2025.
16. S. Filali-Zegzouti, O. Banouar, and M. Benslimanem, *On hyperparameter optimization of generative models in recommendation systems: A comparative study, and applications*, In Optimization and Learning, Springer Nature Switzerland, pp. 34–50, 2026.
17. Y. Koren, R. Bell, and C. Volinsky, *Matrix factorization techniques for recommender systems*, Computer, vol. 42, no. 8, pp. 30–37, 2009.
18. J. Bobadilla, J. Dueñas-Lerín, F. Ortega, and A. Gutierrez, *Comprehensive evaluation of matrix factorization models for collaborative filtering recommender systems*, International Journal of Interactive Multimedia and Artificial Intelligence, 2024.
19. W. Song, and C. Wang, *Hybrid recommendation based on matrix factorization and deep learning*, In Proceedings of the 4th International Conference on Big Data Engineering, pp. 81–85, 2022.
20. A. Sami, W. E. Adrousy, S. Sarhan, et al., *A deep learning based hybrid recommendation model for internet users*, Scientific Reports, vol. 14, pp. 29390, 2024.
21. Z. Elmourabit, and O. Banouar, *GAN based approaches for self-supervised segmentation: A comparative study*, Statistics, Optimization and Information Computing, vol. 12, pp. 646–659, 2024.
22. B. Lim, S. Zohren, and S. Roberts, *Data augmentation techniques in time series domain: A survey and taxonomy*, Neural Computing and Applications, vol. 35, pp. 10123–10145, 2023.
23. G. Douzas, F. Bacao, J. Fonseca, and M. Khudinyan, *A survey on GAN techniques for data augmentation to address the imbalanced data issues in credit card fraud detection*, Machine Learning and Knowledge Extraction, vol. 5, no. 1, pp. 304–329, 2023.
24. D. Chae, J. Kang, S. Kim, and J. Lee, *CFGAN: A generic collaborative filtering framework based on generative adversarial networks*, In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 137–146, 2018.

25. D. Chae, J. A. Shin, and S. Kim, *Collaborative adversarial autoencoders: An effective collaborative filtering model under the GAN framework*, IEEE Access, vol. 7, pp. 37650–37663, 2019.
26. E. Dervishaj, and P. Cremonesi, *GAN-based matrix factorization for recommender systems*, In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, pp. 1373–1381, 2022.
27. O. Banouar, and S. E. Filali Zegzouti, *WGANMF-DO: Matrix factorization with WGAN-GP and dual objective optimization for accuracy and diversity in recommender systems*, In Mathematical Modeling for Data Science, Springer, Cham, 2025.
28. S. Filali-Zegzouti, O. Banouar, and M. Benslimane, *Classification of recommender systems using deep learning based generative models*, In Proceedings of the Statistics and Data Science Conference, 2023.
29. O. Banouar, and S. Raghay, *Pattern-based recommender system using nuclear norm minimization of three-mode tensor and quantum fidelity-based K-means*, International Journal of Pattern Recognition and Artificial Intelligence, vol. 38, no. 06, 2024.
30. R. Lara-Cabrera, Á. González-Prieto, F. Ortega, and J. Bobadilla, *Evolving matrix-factorization-based collaborative filtering using genetic programming*, Applied Sciences, vol. 10, no. 2, pp. 675, 2020.
31. M. Rezaei, and R. Boostani, *Using the genetic algorithm to enhance nonnegative matrix factorization initialization*, Expert Systems, vol. 31, no. 3, pp. 213–219, 2014.
32. Z. A. Khan, N. I. Chaudhary, T. A. Khan, U. Farooq, C. M. A. Pinto, and M. A. Z. Raja, *Enhanced fractional prediction scheme for effective matrix factorization in chaotic feedback recommender systems*, Chaos, Solitons & Fractals, vol. 176, pp. 114109, 2023.
33. Y. Wang, M. Gao, X. Ran, J. Ma, and L. Y. Zhang, *An improved matrix factorization with local differential privacy based on piecewise mechanism for recommendation systems*, Expert Systems with Applications, vol. 216, pp. 119457, 2023.
34. P. Zhang, H. Sun, Z. Zhang, X. Cheng, Y. Zhu, and J. Zhang, *Privacy-preserving recommendations with mixture model-based matrix factorization under local differential privacy*, IEEE Transactions on Industrial Informatics, vol. 21, no. 7, pp. 5451–5459, 2025.
35. D. Chae, J. Kang, S. Kim, and J. Choi, *Rating augmentation with generative adversarial networks towards accurate collaborative filtering*, In Proceedings of The World Wide Web Conference, pp. 2616–2622, 2019.
36. Q. Wang, H. Yin, H. Wang, Q. V. H. Nguyen, Z. Huang, and L. Cui, *Enhancing collaborative filtering with generative augmentation*, In Proceedings of the 25th ACM SIGKDD International Conference, pp. 548–556, 2019.
37. H. Yu, C. Hsieh, S. Si, and I. S. Dhillon, *Parallel matrix factorization for recommender systems*, Knowledge and Information Systems, vol. 41, pp. 793–819, 2014.
38. A. M. A. Al-Sabaawi, H. Karacan, and Y. E. Yenice, *A novel overlapping method to alleviate the cold-start problem in recommendation systems*, International Journal of Software Engineering and Knowledge Engineering, vol. 31, no. 09, pp. 1277–1297, 2021.
39. S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, *BPR: Bayesian personalized ranking from implicit feedback*, arXiv preprint arXiv:1205.2618, 2012.
40. O. Banouar, S. Mohaoui, and S. Raghay, *Collaborating filtering using unsupervised learning for image reconstruction from missing data*, EURASIP Journal on Advances in Signal Processing, vol. 2018, no. 72, pp. 1–13, 2018.
41. X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*, arXiv preprint arXiv:2002.02126, 2020.
42. R. Hassanzadeh, V. Majidnezhad, and B. Arasteh, *A novel recommender system using light graph convolutional network and personalized knowledge-aware attention sub-network*, Scientific Reports, vol. 15, p. 15693, 2025.
43. J. Chen, J. Zhou, and L. Ma, *GNNCL: A Graph Neural Network Recommendation Model Based on Contrastive Learning*, Neural Processing Letters, vol. 56, p. 45, 2024.
44. K. Kim, and H. Ahn, *A recommender system using GA K-means clustering in an online shopping market*, Expert Systems with Applications, vol. 34, no. 2, pp. 1200–1209, 2008.
45. J. Bobadilla, F. Ortega, A. Hernando, and J. Alcalá, *Improving collaborative filtering recommender system results and performance using genetic algorithms*, Knowledge-Based Systems, vol. 24, no. 8, pp. 1310–1316, 2011.
46. J. Xiao, M. Luo, J. Chen, and J. Li, *An item based collaborative filtering system combined with genetic algorithms using rating behavior*, In Advanced Intelligent Computing Theories and Applications, ICIC 2015, pp. 453–460, 2015.
47. M. Salehi, *Latent feature based recommender system for learning materials using genetic algorithm*, Information Systems & Telecommunication, vol. 2, no. 3, pp. 137–144, 2014.
48. Y. B. Jia, Q. Q. Ding, D. L. Liu, J. F. Zhang, and Y. L. Zhang, *Collaborative filtering recommendation technology based on genetic algorithm*, Applied Mechanics and Materials, vol. 599, pp. 1446–1452, 2014.
49. L. Gang, H. Chunling, and C. Shengbing, *Research on recommender system based on ontology and genetic algorithm*, Neurocomputing, vol. 187, pp. 92–97, 2016.
50. B. Alhijawi, and Y. Kilani, *A collaborative filtering recommender system using genetic algorithm*, Information Processing & Management, vol. 57, no. 6, pp. 102310, 2020.
51. Y. Kilani, A. F. Otoom, A. Alsarhan, and M. Almaayah, *A genetic algorithms-based hybrid recommender system of matrix factorization and neighborhood-based techniques*, Journal of Computational Science, vol. 28, pp. 78–93, 2018.
52. T. Elansari, M. Ouanan, and H. Bourray, *Mixed radial basis function neural network training using genetic algorithm*, Neural Processing Letters, vol. 55, no. 8, pp. 10569–10587, 2023.
53. K. E. Permana, *Comparison of user based and item based collaborative filtering in restaurant recommendation system*, Mathematical Modelling of Engineering Problems, vol. 11, no. 7, pp. 1922–1928, 2024.
54. F. M. Harper, and J. A. Konstan, *The MovieLens datasets: History and context*, ACM Transactions on Interactive Intelligent Systems, vol. 5, no. 4, pp. 1–19, 2015.
55. I. Cantador, P. Brusilovsky, and T. Kuflik, *Second workshop on information heterogeneity and fusion in recommender systems*, In Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 387–388, 2011.
56. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, *Item-based collaborative filtering recommendation algorithms*, In Proceedings of the 10th International Conference on World Wide Web, pp. 285–295, 2001.
57. X. Su, and T. M. Khoshgoftaar, *A survey of collaborative filtering techniques*, Advances in Artificial Intelligence, vol. 2009, pp. 1–19, 2009.

58. M. Ferrari Dacrema, S. Boglio, P. Cremonesi, and D. Jannach, *A troubling analysis of reproducibility and progress in recommender systems research*, ACM Transactions on Information Systems, vol. 39, no. 2, pp. 1–49, 2021.
59. M. Ferrari Dacrema, P. Cremonesi, and D. Jannach, *Are we really making much progress? A worrying analysis of recent neural recommendation approaches*, In Proceedings of the 13th ACM Conference on Recommender Systems, pp. 101–109, 2019.
60. S. Nokhwal, S. Nokhwal, S. Pahune, and A. Chaudhary, *Quantum generative adversarial networks: Bridging classical and quantum realms*, arXiv preprint arXiv:2312.09939, 2023.