International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I10P121

Volume 74 Issue 10, 288-307, October 2026
© 2026 Seventh Sense Research Group®

Original Article

A Deep Attribute-Induced Collaborative Filtering Recommendation Model with Attribute-Aware Attention and Item-Priority Modelling

Amritha P[1], Rajkumar KK[2]

[1,2]*Department of Information Technology, Kannur University, Kerala, India.*

[1]*Corresponding Author: amritha@chintech.ac.in*

Received: 25 May 2026 Revised: 12 September 2026 Accepted: 18 September 2026 Published: 27 October 2026

Abstract - *The growth of digital applications has increased information overload, causing difficult for users to find appropriate content. Recommendation systems address this issue by generating customized recommendations based on individual user interests. However, most collaborative and hybrid recommendation methods mainly depend on the rating matrix and dot-product-based similarity computation, which captures only low-order linear interactions. Apart from that, when only ratings are available, collaborative filtering struggles to distinguish between limited data availability and genuine cold-start scenarios. To address these limitations, this paper introduces a deep recommendation framework, termed Deep Attribute-Induced Collaborative Filtering (DeepACF). This unified neural recommendation framework is designed to improve personalized recommendations by jointly modeling user information, item information, contextual attributes, and item-priority features within a single architecture. The proposed model consists of input, embedding, attention, and prediction modules. In the embedding layer, user IDs, item IDs, user-side features, item-side features, and contextual attributes are transformed into dense latent representations through dedicated encoders. In addition, a learned item-priority encoder is introduced to capture item age, popularity, cold-item status, and average rating tendency. These encoded representations are further refined through feature-level attention, which allows the model to allocate varying importance to the available features according to the user-item interaction context. The final representations are then fused and passed through a multilayer perceptron to generate the final predicted score. Experimental results on the Movie-Lens datasets show that DeepACF achieves superior rating prediction accuracy, obtaining RMSE/MAE values of 0.8376/0.6643 and 0.8081/0.6418, respectively. For Top-N recommendation, the model achieves competitive ranking performance, with Hit Rate/NDCG values of 0.7801/0.5504 on Movie-Lens 100K and 0.7985/0.6104 on Movie-Lens 1M.*

Keywords - *Recommendation system, Collaborative filtering, Contextual attributes, Attention, Embedding.*

1. Introduction

With the continuous evolution of technology-based communication and information systems, the volume of data generated and transmitted over the Internet has increased. This continuous growth of digital information has led to the problem of information overload. This problem creating challenges for users in locating information that is relevant to their interests [1, 2]. Recommendation systems offer an effective way to overcome this challenge by filtering large collections of information and presenting personalized items to users. Applications like Amazon, Alibaba, Google, Netflix, and YouTube uses recommender systems to provide relevant products, services, and digital content [3]. Recommendation systems are also broadly adopted in diverse fields, such as education, healthcare, agriculture, tourism, digital libraries, and entertainment for identifying suitable learning resources, medical information, agricultural content, travel destinations, books, research articles, and movies [3, 4]. Conventional recommender systems mainly depend on Collaborative Filtering (CF) and Content-Based Filtering (CBF). The first one analyzes the ratings of a group of users to estimate the interests and choices of a specific user. It is based on the assumption that users who provide similar ratings to items are expected to share similar interests and preferences [5, 6]. However, content-based filtering generates recommendations by matching item characteristics with the interests of users. So, this approach analyzes item characteristics to suggest items resembling to those previously preferred by the user [7].

Although collaborative filtering has been utilized across various recommendation systems, its performance is dependent on the amount of available user-item interaction data. Challenges associated with CF are data sparsity and the cold-start problem. User preferences in CF are commonly

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

represented through a rating matrix [5]. However, users generally interact with a limited number of the available items, leading to a matrix with very limited entries. Such limited interaction information complicates the process of identifying reliable relationships among users and items and may reduce recommendation accuracy. The cold-start problem arises when newly introduced users or items have little or no prior interaction data [6]. A new user provides insufficient information for the model to generate reliable recommendations, while a new item may receive limited exposure because no historical interaction data is available [8, 9]. Many approaches have been developed to overcome these limitations, neighborhood models [10, 11], matrix factorization methods [5] and machine learning-driven recommendation methods [11, 12].

Hybrid methods have also combined matrix factorization with neighborhood information, content features, and other auxiliary information to improve recommendation performance [6, 7]. Factorization machines further extend latent factor modelling by estimating ratings among input features [12]. However, conventional latent-factor approaches generally depend on inner-product operations, which may have limited ability to represent intricate and nonlinear interactions between users, items, and auxiliary features [14, 15].

Deep Learning-driven recommendation systems have subsequently been introduced to learn more complex feature interactions through multiple nonlinear transformations. These methods provide greater flexibility than traditional latent-factor models and allow heterogeneous information to be represented through learned embeddings [15]. The performance of a recommendation model relies not only on its capability to learn nonlinear user-item relationships but also on the amount and additional/side information available to the model [16].

When interaction information or ratings is limited, for sparse or newly introduced items, depending on collaborative signals, it may result in insufficient item representation. This motivates the inclusion of additional user, item, and contextual information that can provide useful evidence even when historical interactions are limited [17, 18]. User preferences are also dynamic rather than completely static. A user's interest varies based on the situation in which an interaction occurs. For example, a user who is interested in action movies may choose comedy movies during weekends or documentaries while travelling. This kind of changes are influenced by contextual factors, including time, location, or other such conditions. Static user characteristics, such as age and gender, and item-specific attributes can also contribute to the recommendation process. Therefore, incorporating contextual and auxiliary attributes along with collaborative information can provide a broader representation of user preferences and item attributes. Such information can be particularly valuable when the interaction matrix contains insufficient information about certain users or items [18].

Existing recommendation approaches have investigated nonlinear interaction modelling, side-information integration, context-aware recommendation, feature embeddings, and attention-based mechanisms. These developments have improved recommender systems to represent complex user preferences. However, an important challenge was effectively combining heterogeneous recommendation signals within a unified model, particularly when item interaction information is sparse. User attributes, item attributes, interaction context, and item-specific characteristics do not necessarily contribute equally to every recommendation decision.

Moreover, characteristics such as item age, popularity, cold-item status, and rating tendency may provide useful information about the importance and current state of an item, but these factors are not always considered together with contextual and collaborative representations. Hence, there is a need for a recommendation framework that jointly represents these heterogeneous attributes and adaptively determines their relevance while learning complex user-item relationships.

To address the research gap, this study introduces a recommendation framework based on deep learning called Deep Attribute-Induced Collaborative Filtering (DeepACF). The proposed model jointly utilizes collaborative information, user attributes, item attributes, contextual information, and item-priority features within a unified neural architecture. Dense representations of users, items, and auxiliary attributes are learned through embedding layers, while contextual information is derived from interaction-related features.

In addition, an item-priority representation is introduced by considering item age, popularity, cold-item status, and average rating tendency to provide additional information about individual items, particularly when their interaction history is limited. An attention mechanism is incorporated to adaptively determine the relative importance of the available attributes according to the interaction being modelled. Through the joint use of deep feature learning, contextual information, attribute-aware attention, and item-priority modelling, DeepACF aims to improve personalized recommendation performance and provide greater robustness under sparse-item and item cold-start conditions.

The novelty of this research is the combined modelling of heterogeneous recommendation information. DeepACF combines user and item representations with auxiliary attributes, interaction context, and item-priority information and uses an attention mechanism to determine their relative contribution to recommendation generation. The item-priority component further supplements collaborative information with characteristics related to item recency, popularity, cold-start status, and rating behaviour. The performance of the

proposed framework is assessed under conventional and cold-start recommendation settings to examine its ability to provide reliable recommendations when interaction information is limited. The structure of the remaining sections of the article is as follows. Section II reviews previous studies and discusses commonly used recommendation approaches. Section III presents the materials and methods, the proposed framework, datasets, experimental settings, parameter configurations, and baseline methods used for performance comparison. Section IV reports the results and discusses the main observations.

2. Literature Review

Studies have shown that neural network-based approaches can improve the performance of recommender systems by learning nonlinear relationships between users and items. Many deep learning-based recommendation models utilize auxiliary information, such as textual descriptions, audio and video content, and item metadata, to improve user and item representations [19]. To alleviate data sparsity, researchers have also explored additional sources of information, including user reviews, comments, social information, and item attributes, to obtain a more comprehensive representation of user preferences [16, 17]. Other studies combine neural networks with traditional recommendation techniques or extend conventional methods through deep neural architectures to improve their representation capability [20]. In real-world applications, diverse forms of structural and contextual information, such as social networks, item characteristics, interaction context, and review data, may also be available. Effectively incorporating such auxiliary information can therefore improve recommendation quality, particularly when user-item interaction data is limited [18].

Li et al. (2020) [21] developed a neural network with adaptive attention for social recommendations, where social-network information and user-friend relationships are represented through neural embeddings. An adaptive attention is used to determines varying influence of friends on user preferences. This approach distinguishes the contribution of different social connections and improves recommendation performance when interaction data is sparse. However, its recommendation process largely depends on social influence, making its applicability on the availability of reliable social-network information. This limitation motivates the development of recommendation approaches that can incorporate diverse auxiliary information without depending heavily on explicit social connections [21].

Kiran et al. (2020) [22] developed DNNRec, is a hybrid recommender engine that uses deep neural architectures. Their system enhances predictive accuracy by joining user and item metadata with learned vector embeddings inside a deep neural network. This architecture captures complex non-linear relationships, enhancing recommendation performance when user or item data is scarce. The advantage of the model is the effective use of auxiliary information together with deep feature learning. However, the model does not explicitly determine the relative importance of different auxiliary attributes according to the interaction context. This limitation provides scope for models that can adaptively weight heterogeneous features while generating personalized recommendations [22].

Chen and Pan (2021) [23] introduced a model to improve the deep collaborative filtering. The method organizes recommendations as a hierarchical filtering process, where irrelevant items are progressively removed before the final recommendation stage. A knowledge-distillation-based training strategy is also used, where a trained collaborative filtering model acts as a teacher and transfers its learned knowledge to guide the training of the hierarchical decision network. However, the method mainly focuses on accelerating the recommendation process and relies on historical user-item interactions, with less consideration of additional contextual and attribute-level information. This limitation indicates the need for models that can combine representation learning with user, item, and contextual features [23].

Harshvardhan et al. (2022) [24] developed a recommendation model that considers temporal information. The approach uses a Restricted Boltzmann Machine model and uses temporal information as side information. The model is strong in capturing the time-dependent changes of the user preferences while also considering missing information via learning of latent features. The model, however, only takes into account rating and temporal information and while overlooking other user and item-specific attributes that might also influence preferences. Such a drawback points to the importance of considering temporal context and a larger set of auxiliary attributes for more personalized recommendations [24].

Wang et al. (2022) [25] introduced a deep collaborative filtering framework that makes use of temporal information for modelling changing user preferences over time. The method uses an attention mechanism that captures changes over time to estimate short-term preferences from recently interacted items and their interaction times, which are then combined with the user's long-term interests. Deep Matrix Factorization and a multilayer perceptron are further employed to learn high-order user-item interactions and predict recommendation scores. The key benefit of this method is that it can capture both changing user interests and complex user-item relationships. However, the model mainly represents preference changes through temporal interaction behaviour and gives limited consideration to other user and item-related attributes [25].

Liu et al. (2023) [26] created a recommendation model based on neural network-based techniques. The meta-learning part of the model helps to learn the preferences of a new user

from only a few available interactions, while the attention mechanism learns different weights for user-item interactions to better capture the user's diverse interests. A major strength of this approach is that it can learn personalized preferences from a small amount of available data, thereby enhancing recommendation accuracy. However, the model mainly concentrates on rapid preference adaptation from limited interactions and gives less attention to contextual variations and item-specific priority characteristics. This limitation leads to models that incorporate richer user, item, and interaction-related information to improve cold-start performance [26].

Lin et al. (2024) [27] proposed NCF-MS, a neural approach that uses multiple sources of information for recommendation. The model uses an Autoencoder to learn latent features from user profiles and service attributes, while a Multilayer Perceptron (MLP) integrates these features with collaborative information to predict user-service interactions. One of the main benefits of this approach is that auxiliary information adds to sparse interaction records and improves recommendation accuracy. However, the model mainly relies on user profiles and service attributes as additional attributes, leaving other dynamic and contextual factors. This limitation indicates the importance of a wide range of interaction-related information [27].

Bhatia (2024) [28] proposed DLSF, a deep learning and semantic fusion-based recommendation framework that consists of three modules. Module 1 applies collaborative filtering with deep latent representations, Module 2 uses semantic information through content-based filtering, and Module 3 generates recommendations for new items without rating history. The system then provides the final recommendation by combining the normalized scores of the applicable modules using a weighted average, while item popularity is also considered when selecting the contribution of collaborative and semantic information.

One of the strengths of DLSF is its ability to employ different recommendation strategies according to the availability and nature of item information. However, the fusion process is mainly governed by predefined module-level rules and weights, rather than dynamically learning the importance of individual attributes for each interaction. This limitation creates scope for more adaptive models that can jointly learn and weight heterogeneous features during recommendation [28].

Behera et al. (2024) [29] come up with a hybrid collaborative recommendation that combines matrix factorization with XGBoost for movie recommendation. The framework is organized into two key stages. The first stage, matrix factorization, estimates an initial rating score from the rating matrix. This prediction is merged with additional features, including similarities between users and items, along with the overall average rating. This is supplied to XGBoost to generate the final rating prediction. The advantage of HCFMR is, it combines matrix factorization with the nonlinear prediction capability of gradient boosting. However, the model depends on interaction-derived and manually constructed features and does not incorporate user, item, or contextual attributes. This limitation motivates the use of models that can automatically learn representations from heterogeneous auxiliary information while capturing complex user-item relationships [29].

Zou et al. (2024) [30] proposed a knowledge graph-based recommendation to capture multiple user intents and reduce irrelevant information in knowledge graphs. The model contains two major components. The first employs a graph transformer to learn global user intents from rating matrix and knowledge-graph information, while the second uses intent-guided knowledge sampling and local-global contrastive learning to reduce the influence of noisy knowledge. One useful feature of KGTN is that it can represent diverse user interests while filtering irrelevant knowledge during representation learning. However, the approach relies on the construction and processing of a knowledge graph and introduces additional complexity through graph-transformer and contrastive-learning components. This creates scope for recommendation models that can capture heterogeneous user and item characteristics directly from available attributes without requiring an external knowledge graph [30].

Remadnia et al (2025) [31] developed a method using Collaborative and Content Filtering. In the collaborative part, latent preferences are modelled by learning user-item embeddings and bias terms are added to model individual user and item tendencies. The content-based component applies regression techniques to user-profile information to support recommendations for new users. The system brings together the outputs of the collaborative filtering and content-based models, giving each one a specific weight to produce the final recommendation score. One of the main benefits of this approach is that it can use both interaction patterns and additional attributes to minimize the impact of sparsity and cold-start conditions. However, the contribution of the two components is controlled through a predefined weighting factor rather than being adjusted according to the relevance of individual attributes for each interaction [31]. The summary of the existing works is listed in Table 1.

Recent deep and hybrid recommendation frameworks have improved the modelling of nonlinear user-item interactions and have enhanced recommendation accuracy through the use of auxiliary information. However, several existing approaches focus on either predictive performance, computational efficiency, or cold-start conditions, while the joint use of heterogeneous user, item, and contextual information remains limited. In addition, the contribution of different auxiliary attributes varies across individual user-item interactions, yet many existing models rely on fixed fusion

strategies. This highlights the need for a recommendation framework that can effectively integrate diverse auxiliary information and dynamically determine the importance of individual attributes. Based on these observations, this work proposes a Deep Attribute-Induced Collaborative Filtering (DeepACF), which uses attribute-level embeddings and an attention mechanism to capture contextual relevance, heterogeneous feature contributions, and complex user-item interactions.

3. Materials and Methods

3.1. Overview

This section explains the proposed Deep Attribute-Induced Collaborative Filtering (DeepACF) framework. The model is built around four main components: the input, embedding, attention, and prediction layers. These layers are responsible for the following functions; they are input representation, feature encoding, context-aware attention, feature interaction and prediction. In the input stage, user information, item information, and interaction-specific contextual attributes are extracted from the dataset. These inputs are first transformed into a representation suitable for neural learning. Since the dataset contains heterogeneous attributes, including categorical, numerical, and contextual features, they are encoded into dense latent representations before prediction. The proposed model employs trainable embedding and neural encoding functions to represent user-related features, item-related features, and contextual attributes in a low-dimensional dense form. User and item identifiers are mapped through embedding layers, while auxiliary side and contextual features are encoded through learnable transformations. To enhance representation learning, a feature-level context-aware attention mechanism is introduced, where the current user-item pair guides the selective weighting of informative user-side, item-side, and contextual features.

To further improve the representation of items with limited historical interactions and to incorporate temporal relevance, the proposed model introduces an additional recency/cold-start-aware item descriptor within the embedding layer. For every item i, the item age, popularity, cold-item status, and item mean rating are computed as auxiliary item-priority features. The attended representations are then combined through feature interaction operations, including concatenation, element-wise product and dot-product similarity, before being passed to a multilayer perceptron for final rating prediction. All components of the framework, embeddings, feature encoders, attention modules, and prediction layers, are optimized jointly through forward and backward propagation. Figure 1 provides an overview of the proposed model architecture.

In the suggested recommendation model, user-side information, item-side information, and contextual attributes play an important role in improving prediction accuracy beyond conventional rating matrix modelling. In this work, the auxiliary data includes user attributes such as age, gender, and occupation; item attributes such as genre and year; and contextual or temporal information such as weekday and hour of the day. These heterogeneous features are first transformed into dense low-dimensional latent representations through trainable embedding and feature encoding layers. The proposed model employs a feature-level context-aware attention, in this mechanism, the current user-item pair guides the selective weighting of informative auxiliary features.

As a result, the model can dynamically determine which user-side, item-side, and temporal/contextual features are more relevant for a given interaction. For example, temporal features may receive higher attention when modelling recent or time-sensitive preferences, whereas demographic attributes such as age or occupation may contribute more strongly to long-term user preference patterns.

3.2. Problem Formulation

This section describes the proposed Deep Attribute-Induced Collaborative Filtering (DeepACF) framework. Given recommendation problem, let U=$\{u_1, u_2 \ldots u_M\}$ refer to user sets and I=$\{i_1, i_2 \ldots i_N\}$ Indicate the item sets. User ratings are represented through a rating matrix $R \in R^{MxN}$, with each value r_{ui} corresponds to the score given by the user u on item i.

In addition to user and item identities, the model also incorporates auxiliary information. Let x_u denote the user attribute vector, x_i denote the item attribute vector, and x_{ui} denote the interaction/context feature vector. The objective is to learn a nonlinear prediction function; the predictive model can be formulated as:

$$\hat{r}_{ui} = f(u, i, x_u, x_i, x_{ui}; \Theta) \quad (1)$$

where $\hat{r}_{ui}$ refers to the estimated rating, and Θ denotes all trainable parameters.

3.3. Input Layer

In this implementation, multiple input components are used to capture both collaborative and contextual information. These inputs are extracted from the dataset and pre-processed using categorical encoding to convert raw attributes into integer indices suitable for embedding. The input layer receives the raw information associated with each user-item interaction. For a given pair (u, i), the model takes the user ID u, item ID i, the feature vector that represents the user x_u, the one that represents the item x_i and interaction/context features x_{ui} as inputs.

These raw inputs provide the identity information and auxiliary attributes required for personalized recommendations.

Table 1. Literature review summary of recommendation models with methodological approaches, strengths, and research gaps

Author / Year	Methodology	Advantage	Research Gap
Li et al. (2020) [21]	Neural representations of social relationships with adaptive attention	Captures varying social influence.	Depends on the availability of reliable social-network information; its applicability limited when explicit social relationships are unavailable.
Kiran et al. (2020) [22]	Integrates user and item side information, embedding a deep neural network	Auxiliary information with deep feature learning.	Side information is integrated into the deep architecture, but the relative importance of heterogeneous attributes is not considered.
Chen and Pan (2021) [23]	Hierarchical decision network and pretrained Collaborative Filtering	Reduces the computational cost of deep CF.	The objective is faster recommendation and candidate filtering rather than enriching preference modelling with heterogeneous user, item, and contextual information.
Harshvardhan et al. (2022) [24]	Unsupervised Restricted Boltzmann Machine	Temporal information and captures changes in user preference.	Considers rating and temporal information; other user characteristics, item attributes, and interaction-related factors receive limited attention.
Wang et al. (2022) [25]	Attention-Driven User Modeling with Deep Collaborative Filtering	Captures dynamic preference and nonlinear relationships.	Dynamic preference is mainly derived from temporal interaction behaviour, while user and item attributes are less explicitly incorporated.
Liu et al. (2023) [26]	Combines meta-learning with an attention mechanism.	Rapid preference adaptation with sparse data.	Concentrates on rapid preference adaptation from limited interactions; broader contextual and item-specific priority information is not in focus.
Lin et al. (2024) [27]	Stacked Denoising Autoencoder neural collaborative filtering	Auxiliary user and Item information with sparse data.	Focuses on profile and service attributes; dynamic context and interaction-specific adaptive weighting of individual attributes are less explored.
Behera et al. (2024) [29]	Matrix factorization was subsequently processed using XGBoost.	Latent-factor modelling with the nonlinear prediction.	Interaction-derived features rather than jointly learning richer heterogeneous user, item, and contextual representations.
Bhatia (2024) [28]	Deep collaborative filtering, Semantic content-based, and Popularity-based.	Multi-sources to address sparsity, scarcity, and cold-start.	Fusion is performed through predefined module priorities and normalized score weighting rather than dynamically learning.
Zou et al. (2024) [30]	Graph transformer intent-guided knowledge sampling and local-global contrastive denoising.	Multiple user intents and reducing irrelevant knowledge-graph data	Requires construction and processing of a knowledge graph and introduces additional graph-transformer and contrastive-learning complexity.
Remadnia et al. (2025) [31]	Collaborative Filtering and Embedding Based Deep Learning	Combines interaction information and content attributes.	Uses a predefined fusion weight between collaborative and content scores instead of adapting feature contributions.

3.4. Embedding Layer

This layer transforms the inputs to dense latent vectors suitable for neural learning. Initially, the discrete user ID and item ID are mapped into trainable embedding vectors. These embeddings represent latent embeddings of users and items.

The user identifier embedding $e_u = E_U(u)$, the item identifier embedding $e_i = E_I(i)$ where E_U and E_I are the learned representation matrices of users and items. Similarly, the user-side, item-side, and contextual features are encoded into dense latent representations through neural feature encoders. These embeddings are:

$$e_{x_u} = \phi_u(x_u),$$

$$e_{x_i} = \phi_i(x_i) \text{ and,}$$

$$e_{x_{ui}} = \phi_{ui}(x_{ui})$$

where ϕ_u is the attribute encoder, ϕ_i is the item attribute encoder and ϕ_{ui} is the interaction/context feature encoder. These encoders consist of trainable embedding lookup layers for categorical attributes and dense projection layers for numerical or temporal attributes. Thus, the embedding layer produces latent representations for user identity, item identity, and auxiliary features.

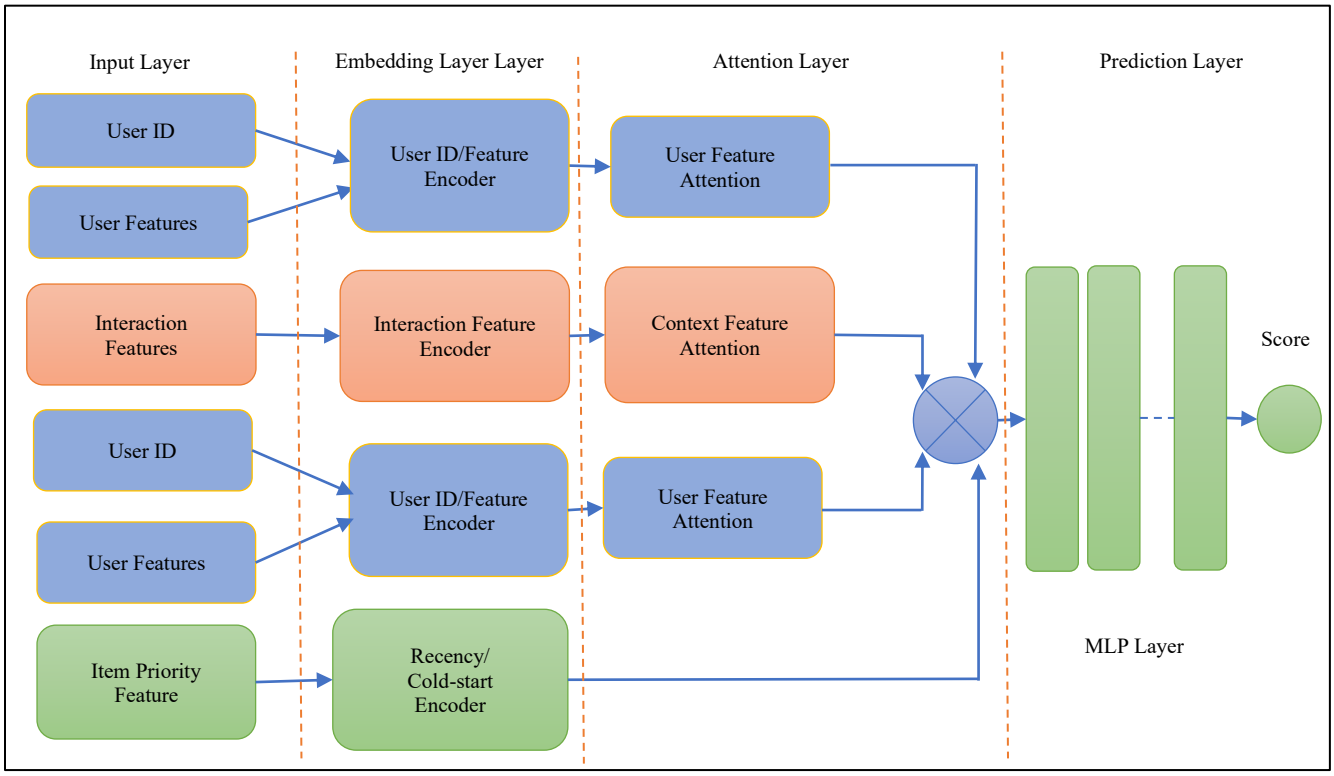


Fig. 1 Proposed architectural diagram of DeepAC

To incorporate time-aware and cold-start-related item characteristics, an additional item-priority feature vector is constructed for each item i. This is a cold-start-aware item descriptor, explicitly models the temporal age, popularity, and cold-start status of each item. First, the item age of item i is defined as:

$$a_i = t_{ui} - y_i \tag{2}$$

where t_{ui} denotes the interaction year and y_i denotes the release year of item i. Thus, a_i measures how old the item is at the time of interaction. A smaller value of a_i indicates a newer item, while a larger value indicates an older item.

The popularity feature of item i is defined as:

$$c_i = \log(1+n_i) \tag{3}$$

where n_i denote the number of interactions associated with item i. This term reflects the popularity of the item. Here we are using a logarithmic transformation to compress the scale of interaction counts and avoid excessively large values.

To identify items with insufficient historical interactions, a cold-item indicator is introduced as:

$$d_i = \begin{cases} 1, & n_i < \tau, \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where τ is a predefined cold-item threshold.

To find the average rating tendency, the item mean rating is defined as:

$$r_i = \frac{1}{n_i} \sum_{u \in U_i} r_{ui} \tag{5}$$

where r_{ui} is the rating in the rating matrix.

Finally, the raw item-priority feature vector is constructed by combining item age a_i, popularity feature c_i, cold-item indicator d_i and average rating tendency r_i as:

$$\bar{r_i} = [a_i \parallel c_i \parallel d_i \parallel r_i] \tag{6}$$

This vector is then projected into a dense latent representation through a trainable nonlinear transformation:

$$e_{r_i} = \phi_r(\bar{r_i}) \tag{7}$$

where ϕ denotes the recency/cold-start embedding function or encoder. Here, the user-feature, item-feature, contextual feature and item priority encoders are implemented as trainable linear projections followed by a ReLU activation. The embedding representation is shown in Figure 2.

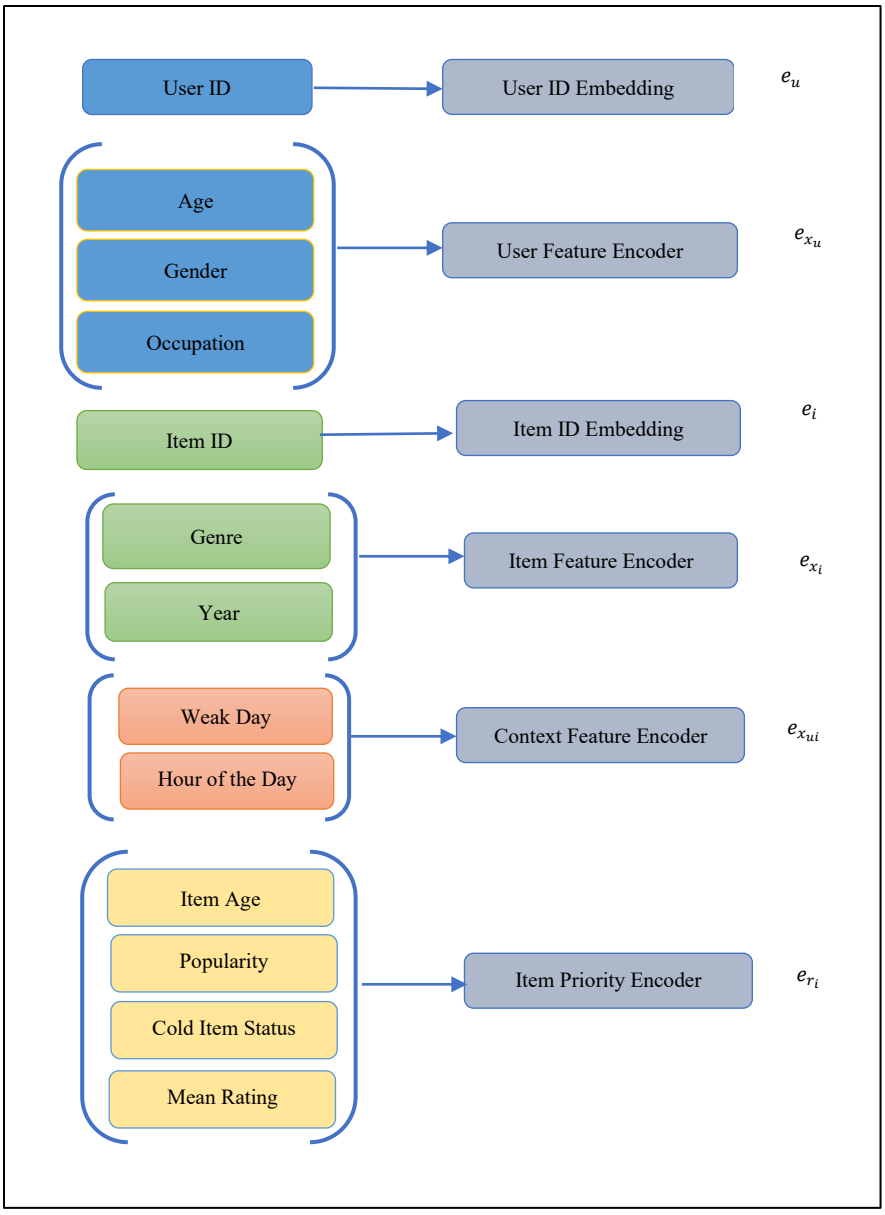


Fig. 2 Embedding representations in DeepACF

3.5. Attention Layer

The attention mechanism dynamically identifies how much weight should be given to different of auxiliary user-side, item-side, and contextual features for each user-item interaction. Rather than all auxiliary features contributing equally, the proposed model uses the current user-item pair as a guiding signal to selectively emphasize the most informative feature components.

To do this, a query vector is first constructed from the user and item identity embeddings as

$$g_{ui}=\psi([e_u \parallel e_i])$$

where e_u and e_u represent the identity embeddings of the user and the item, respectively, and ψ is a learnable projection function. The query vector provides an interaction representation of the current user-item pair and is used to guide attention over the encoded auxiliary features. For user-side features, the attention is defined as:

$$\alpha_u=\text{softmax}\left(W_{au}\tanh\left(W_{ug}g_{ui}+W_{xu}e_{x_u}+b_u\right)\right) \quad (8)$$

where α_u denotes the attention weights assigned to the encoded user feature vector e_{x_u} and, W_{au}, W_{ug} , W_{xu} and b_u are trainable parameters.

After this step, apply the attention weight to user feature vector:

$$e'_{x_u}=\alpha_u \odot e_{x_u} \quad (9)$$

where e'_{x_u} attention-weighted user feature representation. This operation enables the model to adaptively determine which user-side attributes are more relevant for the current interaction.

Similarly, for item-side features, the attention mechanism is defined as:

$$\alpha_i=\text{softmax}\left(W_{ai}\tanh\left(W_{gi}g_{ui}+W_{xi}e_{x_i}+b_i\right)\right) \quad (10)$$

where α_i denotes the attention weights assigned to the encoded item feature vector e_{x_i} and, W_{gi} , W_{xi} and b_i are trainable parameters.

The attention-weighted item feature representation as follows:

$$e'_{x_i}=\alpha_i \odot e_{x_i} \quad (11)$$

Here α_i assigns attention weights to the item feature vector e'_{x_i} producing attention weights assigned item-side representation e_{x_i}. This allows the model to selectively emphasize item characteristics according to the current user-item pair.

Once the weighted user-side and item-side features are obtained, they are combined to construct the final representations of the user and item as follows:

$$p_u = \left[e_u \parallel e'_{x_u}\right] \quad (12)$$

$$q_i = \left[e_u \parallel e'_{x_i}\right] \quad (13)$$

where p_u refers to the final embedding of the user, q_i is the final item embedding, e_u is the user ID embedding, e_i is the item ID embedding, e'_{x_u} denotes attention-weighted user feature representation and e'_{x_i} denote attention-weighted item feature representation.

To model the importance of contextual features, a similar attention mechanism is applied using the fused user and item representations as the conditioning signal. Hence, the contextual attention is defined as:

$$\alpha_{ui}=\text{softmax}\left(W_{ac}\tanh\left(W_{pc}\left[p_u \parallel q_i\right]+W_{xc}e_{x_{ui}}+b_c\right)\right) \quad (14)$$

where W_{ac}, W_{pc}, W_{xc}, b_c are trainable parameters and α_{ui} contextual attention weight. p_u refers final user representation, q_i refers item representation and $e_{x_{ui}}$ indicate the encoded context features.

Finally, attention-weighted encoded contextual feature vector can be computed as:

$$e'_{x_{ui}}=\alpha_{ui} \odot e_{x_{ui}} \quad (15)$$

Where $e'_{x_{ui}}$ denote attention-weighted encoded contextual feature vector and $e_{x_{ui}}$ indicate the encoded context features.

3.6. Prediction Layer

This layer combines the final representations and attention-weighted encoded contextual feature vector. These interaction signals are then passed into a multilayer perceptron to model nonlinear relationships and generate the final predicted rating.

After the embedding and attention stages, the model has already obtained three important latent vectors: the final user representation p_u, the final item representation q_i, the attention-weighted contextual feature representation $e_{x_{ui}}$ and item-priority feature vector e_{r_i} . These vectors are integrated through multiple interaction operations to form a richer joint representation for prediction.

To analyse user-item compatibility from different perspectives, the model constructs the following interaction vector:

$$h_0 = \left[p_u \parallel q_i \parallel (p_u \odot q_i) \parallel e'_{x_{ui}} \parallel e_{r_i}\right] \quad (16)$$

where p_u denote the latent representation of the user, q_i the latent representation of the item, $p_u \odot q_i$ describes their interaction, $e'_{x_{ui}}$ indicate the attention-weighted contextual feature vector and e_{r_i} is the recency/cold-start feature.

The resulting vector h_0 is then passed through the hidden layers of the MLP. The first one is computed as:

$$h_1=\sigma(W_1 h_0+b_1) \quad (17)$$

where W_1 and b_1 are trainable parameters of the model, and σ denotes the nonlinear activation function ReLU.

After passing through an L layer fully connected network:

$$h_l=\sigma(W_l h_{l-1}+b_l) \; l=2,3,\ldots,L \quad (18)$$

where L denotes the total count of hidden layers, and W_l and b_l are the trainable parameters.

Finally, the representation from the final hidden stage is fed into a linear output function to estimate the rating:

$$\hat{r}_{\text{ui}}=\text{W}_o h_L+\text{b}_o \quad (19)$$

Here W_o and b_o indicate trainable parameters.

3.7. Model Training and Optimization

The proposed DeepACF framework was trained for both explicit rating prediction and item ranking. In both settings, the model is learned by jointly optimizing all trainable parameters, including the user and item embedding matrices, feature encoders, attention-layer parameters, and MLP weights and biases.

For explicit recommendation, the model predicts a rating $\hat{r}_{ui}$. The training objective is to reduce the discrepancy between predicted and ground-truth ratings. r_{ui}. For rating prediction, the objective function is formulated as:

$$L_{\text{pred}} = \frac{1}{|\Omega|}\sum_{(\text{u,i})\in\Omega}(r_{\text{ui}} - \hat{r}_{\text{ui}})^2 +\lambda \parallel \Theta \parallel_2^2 \quad (20)$$

where Ω is the total number of observed ratings., r_{ui} is the actual rating, $\hat{r}_{\text{ui}}$ is the predicted score, Θ denotes all learnable model parameters, and λ controls the strength of the regularization term.

For Top-N recommendation, the task is formulated as an implicit-feedback learning problem rather than forecasting exact rating scores. The framework calculates the likelihood of user u engaging positively with item i. In this case, the model first produces a preference score s_{ui} then passes it through a sigmoid function to yield a final probability as:

$$s_{\text{ui}}=\text{W}_o h_L+\text{b}_o \quad (21)$$

$$\hat{y}_{\text{ui}}=\sigma(s_{\text{ui}}) = \frac{1}{1+\text{e}^{-s_{\text{ui}}}} \quad (22)$$

where s_{ui} is the preference score, $\hat{y}_{\text{ui}}$ is the predicted preference probability.

The Top-N recommendation objective is then defined using binary cross-entropy is given by:

$$L_{\text{topN}} = -\sum_{(\text{u,i})\in D}\big(y_{\text{ui}}\log\hat{y}_{\text{ui}} + (1-\hat{y}_{\text{ui}})\log(1 - y_{\text{ui}})\big) +\lambda \parallel \Theta \parallel_2^2 \quad (23)$$

Where D represents the collection of positive and negative user-item pairs used during training, y_{ui} represents the true binary preference label, $\hat{y}_{\text{ui}}$ is the estimated preference probability, Θ denotes all trainable parameters and λ controls the level of regularization.

3.8. Experiments and Settings

This section evaluates the proposed DeepACF model by comparing its performance against several baseline approaches. The subsequent subsections outline the datasets, describe the experimental configuration, and provide a detailed discussion of the obtained results.

3.8.1. Datasets

For evaluation, the proposed work used the MovieLens benchmark datasets [32, 33]. The both datasets contain user-level attribute information, including user ID, age, gender, and occupation.

In addition, item-specific attributes, including movie ID, title, genres, and year of release, are available in these datasets. The summary of the datasets is listed in Table 2.

3.8.2. Evaluation Parameters

The proposed model is assessed for rating prediction using MAE and RMSE as performance measures. Both metrics calculate the difference between estimated and true ratings. Low error indicates better model accuracy. MAE gives the average absolute prediction error, whereas RMSE gives more importance to larger prediction errors because the errors are squared [34]. For Top-K recommendation, two ranking-based evaluation measures are used, they are Hit-Rate and Normalized Discounted Cumulative Gain. Each user has one ground-truth test item, and the recommendation model ranks this item together with a set of unrated candidate items. Hit-Rate HR@K checks whether the actual relevant item is included in the top-K recommended items. A higher value means the relevant item is more likely to appear in the top-K recommendations. NDCG@K takes the item's position in the ranked list into account, with a higher value indicating better ranking performance [34].

3.8.3. Experimental Setup

The deep learning model is developed in Python with the PyTorch framework. All implementations were conducted on a system with Ubuntu 22.04 LTS. The GPU acceleration is provided by an NVIDIA GeForce RTX 3050. Python version 3.9.15 was used for the experimental analysis, and all executions were carried out in a Jupyter Notebook using Anaconda Navigator 2.4.3 as the development environment

The user ID and item ID embedding dimensions were in the range {16, 32, 64}, the feature embedding dimension was varied over {8, 16, 32, 64}, and the attention dimension from {16, 32, 64}. The dropout rate was tested in the range {0.1, 0.2, 0.3}, the different learning rates were tested over {0.0005, 0.001, 0.002}, and the batch size was selected from {128, 256, 512}. Based on the experimental results, the final

configuration was fixed. For implicit ranking, positive interactions are defined as ratings at least 4. For each positive interaction, 4 negative samples are generated during training.

During evaluation, each test instance is ranked against 99 sampled negative items following the standard leave-one-out protocol. The cutoff value for Top-N recommendation is fixed at K value 10. To identify sparse items, the cold-item threshold is set to 5, that is, items with fewer than 5 observed training interactions are treated as cold items.

To evaluate the proposed model in the cold-start scenario, the Movie-Lens dataset is first divided into split into training and testing sets. Next, the training interactions associated with each item are counted. An item is considered a cold item if it has with fewer than five interactions in the training data. The model is trained on all the training data, and the cold-start evaluation is performed only on the test instances corresponding to these cold items. This setting helps analyze how well the model performs for items with very limited historical information.

3.8.3. Baseline Methods

To evaluate the proposed DeepACF model, a set of conventional, latent factor models, neural network models, hybrid models, and ranking-oriented recommendation methods were selected as baselines. USERKNN [35], SVD [35], and NCF [36] were used for both rating prediction and ranking evaluation.

USERKNN is a neighbourhood-based method that identifies users with similar rating patterns and predicts preferences from the ratings of nearest neighbours.

SVD represents users and items through low-dimensional latent factors and estimates preferences from their learned latent relationships. NCF extends collaborative filtering through neural networks and learns nonlinear user-item interactions from user and item embeddings. For rating prediction work, DNNRec [22], AMeLU [26], HCFMR [29], and DLSF [28] are used with the traditional models.

DNNRec is a deep model that combines learned embeddings with user and item side information for capturing non-linear preferences. AMeLU integrates meta-learning with an attention mechanism to improve recommendations for users with only a few available interactions. HCFMR combines matrix factorization with XGBoost, where latent and interaction-derived features are used to improve the rating prediction. DLSF constructs a three-module hybrid framework by incorporating collaborative, semantic and cold-start recommendation strategies and then fuses their normalised scores using weighted fusion. For ranking-based evaluation, Deep-KMeans [37], KGTN [30], and GSetRank [38] were used in addition to the common baselines. Deep-KMeans combines deep preference representation with K-Means clustering to support personalized Top-N recommendation. KGTN uses a knowledge-enhanced multi-intent graph transformer with intent-guided sampling and contrastive learning to obtain informative representations for item ranking. GSetRank is a ranking-oriented recommendation approach that exploits similarity-based preference information to determine the relevance of candidate items and generate Top-N recommendation lists.

4. Results and Analysis

4.1. Comparison with Existing Baseline Approaches

How well the proposed system performs is analyzed in this subsection based on the results obtained during experimentation. Table 3 illustrates the rating-prediction evaluation of DeepACF with the baseline models on the MovieLens datasets using RMSE and MAE, where smaller values mean the predictions are more accurate. On 100K dataset, USERKNN and SVD listed higher errors, with RMSE values of 0.9460 and 0.9380, respectively. The nonlinear preference learning in NCF reduced the RMSE to 0.9030, DNNRec and AMeLU improved the prediction performance to 0.8840 and 0.8821.

Table 2. Statistical overview of the movie-lens datasets

	ML 100K	**ML 1M**
Totlal Users	943 Users	6040 Users
Total Items	1682 Items	3952 Items
Total Ratings	100000	1000209
Total Ratings per User	106.04	165.59
Total Ratings per Item	59.453	253.089
Sparsity	93.7%	95.81%

The hybrid approaches HCFMR and DLSF achieved better results, with DLSF obtaining the best baseline performance of 0.8402 RMSE and 0.6738 MAE. DeepACF achieved the lowest errors of 0.8376 RMSE and 0.6643 MAE, corresponding to improvements of approximately 0.31% and 1.41%, respectively, over DLSF.A similar observation is seen on MovieLens-1M. USERKNN recorded RMSE was 0.8932 and MAE value of 0.7487, while SVD and NCF improved the results through latent-factor and nonlinear interaction modelling. The advanced baseline method HCFMR, which

achieved 0.8356 RMSE and 0.6691 MAE, and DLSF, which achieved the best baseline results of 0.8159 RMSE and 0.6512 MAE. The DeepACF model achieves an RMSE of 0.8081 and an MAE of 0.6418, which is about 0.96% lower RMSE and 1.44% lower MAE than the DLSF model. DNNRec, AMeLU, HCFMR, and DLSF utilized deep learning, attention and auxiliary information and achieve lower prediction errors. DeepACF extends these capabilities by jointly modelling user and item representations together with auxiliary and contextual attributes and by adaptively assigning importance to these features through attribute-aware attention. In addition, item-priority information provides supplementary signals related to item characteristics, which is particularly useful when interaction information is limited.

Table 4 compares the ranking performance of DeepACF with baseline method using Hit-Rate@10 and NDCG@10. High values of the evaluation parameter indicate better recommendation performance. On the 100K dataset, the USERKNN and SVD methods showed low ranking performance, while deep and ranking-oriented models produced improvements in the results.

Deep-KMeans obtained a Hit-Rate of 0.6633 and NDCG of 0.3814, whereas NCF increased HR@10 to 0.7050 and NDCG@10 to 0.4090. KGTN obtained 0.7475 and 0.4398 for HR@10 and NDCG@10, respectively. The best baselines, GSetRank achieved an HR@10 of 0.7740 and the highest baseline NDCG@10 of 0.5563. DeepACF obtained the highest HR@10 of 0.7801, slightly higher probability of retrieving relevant items within the Top-10 list. And its NDCG@10 of 0.5504, which is lower than the 0.5563 obtained by GSetRank. This shows that DeepACF retrieved relevant items, although GSetRank placed them marginally higher in the ranked list on MovieLens-100K.

On 1M, the stronger models again produced better ranking performance. Deep-KMeans obtained a Hit-Rate of 0.7309, and KGTN increased this value to 0.7709 with an NDCG@10 of 0.5307. GSetRank lists the highest HR@10 of 0.8012, which is close by DeepACF with 0.7985. However, DeepACF obtained the highest NDCG@10 of 0.6104, exceeding the 0.6052 achieved by GSetRank. This result suggests that although GSetRank retrieved relevant items for slightly more users, DeepACF is better at placing relevant items ranked near the top of the recommendation list.

The overall results demonstrate that DeepACF is competitive with the strongest ranking-oriented baseline. Its performance can be attributed to the joint modelling of user and item representations, contextual attributes, and item-related information, together with the attribute-aware attention mechanism. Rather than assigning equal importance to all available features, the attention component allows the model to emphasize information that is more relevant to a particular user-item interaction. The item-priority component further introduces signals related to item popularity, recency, rating tendency, and cold-item status.

4.2. Assessment of Model Performance in Cold-Start Conditions

To further examine the performance of the proposed DeepACF model, an additional evaluation was carried out under the item cold-start setting. Here, the cold-start items are defined as items with no prior ratings in the training set. To simulate this condition, approximately 5% of the items were randomly selected and completely removed from the training data, while their interactions were retained only in the test set. This setting makes it possible to evaluate the model on newly introduced items for which no historical interaction information is available during training. The results in Tables 5 to 8 show that the proposed model maintains stable performance under item cold-start conditions on both datasets. Although the cold-start results are consistently lower than the corresponding non-cold-start results, the performance degradation remains moderate across both rating prediction and Top-N recommendation tasks.

On Movie-Lens 100K, the best cold-start rating prediction performance is achieved at with an RMSE of 0.8789 and an MAE of 0.6802, compared with the non-cold-start values of 0.8376 and 0.6644, respectively. Similarly, for Top-N recommendation on Movie-Lens 100K, the best cold-start results are obtained at with Hit Rate@10 of 0.7518 and NDCG@10 of 0.5304, while the corresponding non-cold-start values are 0.7802 and 0.5504.

Table 3. Comparative analysis of prediction accuracy measured by MAE and RMSE

	Movie-Lens 100K		Movie-Lens 1M	
Model	**R.M.S.E**	**M.A.E**	**R.M.S.E**	**M.A.E**
USERKNN	0.9460	0.7690	0.8932	0.7487
SVD	0.9380	0.7370	0.8740	0.7146
NCF	0.9030	0.7150	0.8530	0.6640
DNNRec	0.8840	0.7060	0.8750	0.6880
AMELU	0.8821	0.7277	0.8663	0.6783
HCFMR	0.8450	0.6970	0.8356	0.6691
DLSF	0.8402	0.6738	0.8159	0.6512
DeepACF	0.8376	0.6643	0.8081	0.6418

Table 4. Comparative analysis hit rate and NDCG with state-of-art baseline

	Movie-Lens 100K		Movie-Lens 1M	
Model	**Hit-Rate@10**	**N.D.C.G@10**	**Hit-Rate@10**	**N.D.C.G@10**
USERKNN	0.0300	0.3530	0.0400	0.3530
SVD	0.3580	0.3710	0.3690	0.3880
Deep-KMeans	0.6633	0.3814	0.7309	0.3869
NCF	0.7050	0.4090	0.7050	0.4090
KGTN	0.7475	0.4398	0.7709	0.5307
GSetRank	0.7740	0.5563	0.8012	0.6052
DeepACF	0.7801	0.5504	0.7985	0.6104

A similar result is observed on Movie-Lens 1M. For rating prediction, the best cold-start performance is obtained with an RMSE of 0.8300 and an MAE of 0.6562, whereas the corresponding non-cold-start results are 0.8081 and 0.6418. In the ranking setting, the strongest cold-start performance is achieved with Hit-Rate@10 of 0.7417 and NDCG@10 of 0.5892, compared with 0.7915 and 0.6114 for the non-cold-start case. These results indicate that, although the absence of historical item interactions naturally reduces performance, the proposed model is still able to preserve a high level of recommendation quality under sparse-item conditions. This shows that the learned item-priority feature representation contributes positively to handling newly introduced or previously unseen items.

4.3. Impact of Latent Factor Size

Table 5. Prediction performance under cold-start conditions Movie-Lens 100K

Latent Factor Size	**8**		**16**		**32**		**64**	
	R.M.S.E	M.A.E	R.M.S.E	M.A.E	R.M.S.E	M.A.E	R.M.S.E	M.A.E
Non-Cold-Start	0.8880	0.6883	0.8657	0.6809	0.8376	0.6644	0.8421	0.6741
Cold-Start	0.9204	0.7223	0.9113	0.7127	0.8789	0.6802	0.8838	0.6906

Table 6. Prediction performance under cold-start conditions on Movie-Lens 1M

Latent Factor Size	**8**		**16**		**32**		**64**	
	R.M.S.E	M.A.E	R.M.S.E	M.A.E	R.M.S.E	M.A.E	R.M.S.E	M.A.E
Non-Cold-Start	0.9014	0.6676	0.8677	0.6557	0.8141	0.6498	0.8081	0.6418
Cold-Start	0.9361	0.7033	0.9040	0.6816	0.8643	0.6790	0.8300	0.6562

Table 7. Evaluation of cold-start conditions ranking on Movie-Lens 100K

Latent Factor Size	**8**		**16**		**32**		**64**	
	Hit-Rate	N.D.C.G	Hit-Rate	N.D.C.G	Hit-Rate	N.D.C.G	Hit-Rate	N.D.C.G
Non-Cold-Start	0.7248	0.5210	0.7648	0.5274	0.7802	0.5504	0.7795	0.5441
Cold-Start	0.7011	0.5037	0.7266	0.5096	0.7518	0.5304	0.7460	0.5223

Table 8. Evaluation of cold-start conditions ranking on Movie-Lens 1M

Latent Factor Size	**8**		**16**		**32**		**64**	
	Hit-Rate	N.D.C.G	Hit-Rate	N.D.C.G	Hit-Rate	N.D.C.G	Hit-Rate	N.D.C.G
Non-Cold-Start	0.7261	0.5649	0.7716	0.5877	0.7915	0.6114	0.7910	0.6104
Cold-Start	0.7137	0.5356	0.7297	0.5402	0.7417	0.5892	0.7405	0.5874

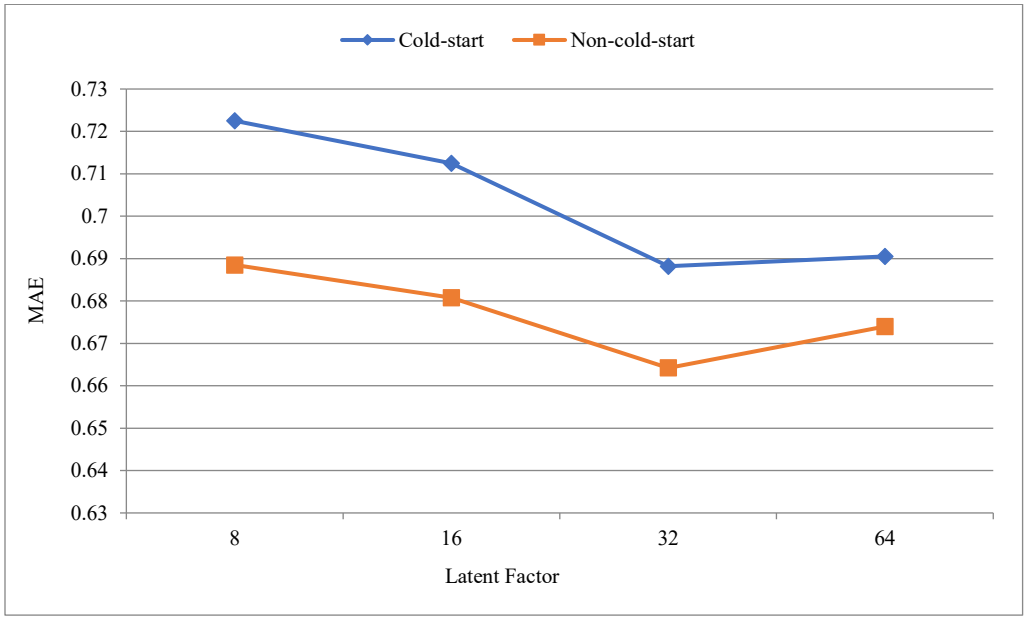
To examine the impact of representation capacity on model performance, the latent factor size was varied across {8, 16, 32, 64}, and the resulting performance was analyzed under both non-cold-start and item cold-start settings. The results in Figures 3 to 6 indicate that the latent factor size has a clear influence on both rating prediction and Top-N recommendation performance.

On Movie-Lens 100K, increasing the latent factor size from 8 to 32 improves both prediction and ranking. For non-cold-start rating prediction, RMSE improves from 0.8880 at latent size 8 to 0.8376 at size 32, and MAE improves from 0.6883 to 0.6644. Under cold-start conditions, RMSE and MAE also improve from 0.9204 and 0.7223 at size 8 to 0.8789 and 0.6802 at size 32.

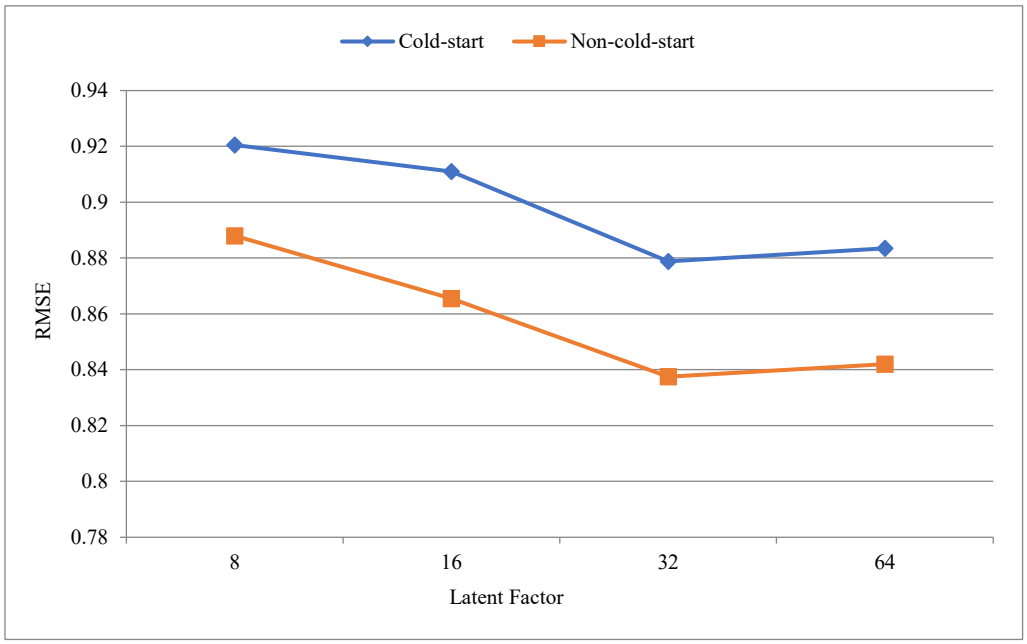
A similar pattern is observed for ranking performance. In the non-cold-start case, Hit-Rate@10 increases from 0.7248 to 0.7802, while NDCG@10 also increases from 0.5210 to 0.5504. Under cold-start conditions, the corresponding values improve from 0.7011 and 0.5037 to 0.7518 and 0.5304. When the latent factor size is further increased to 64, the rating performance slightly degrades, and the ranking performance does not improve consistently, indicating that latent size 32

provides the best balance for the smaller dataset. On Movie-Lens 1M, the influence of latent factor size is different. For rating prediction, performance continues to improve as the latent size increases, with the best non-cold start results obtained at size 64 and the best cold-start results also obtained at size 64.

This suggests that the larger dataset can better support higher-dimensional latent representations for rating estimation. However, for Top-N recommendation, the slight improvement in performance is achieved at latent size 32 rather than 64. Under cold-start conditions, size 32 again, gives small improvements in ranking performance. When the latent size is increased to 64, the ranking performance decreases slightly, showing that a moderate latent size is more suitable for Top-N recommendation. In addition to the above analysis, the graphical results illustrate how the effectiveness of the latent representation changes with the depth of the MLP component. Across the tested latent dimensions, the MLP-4 configuration generally produces the lowest MAE and RMSE on both datasets. This is illustrated in Figures 7 and 8.

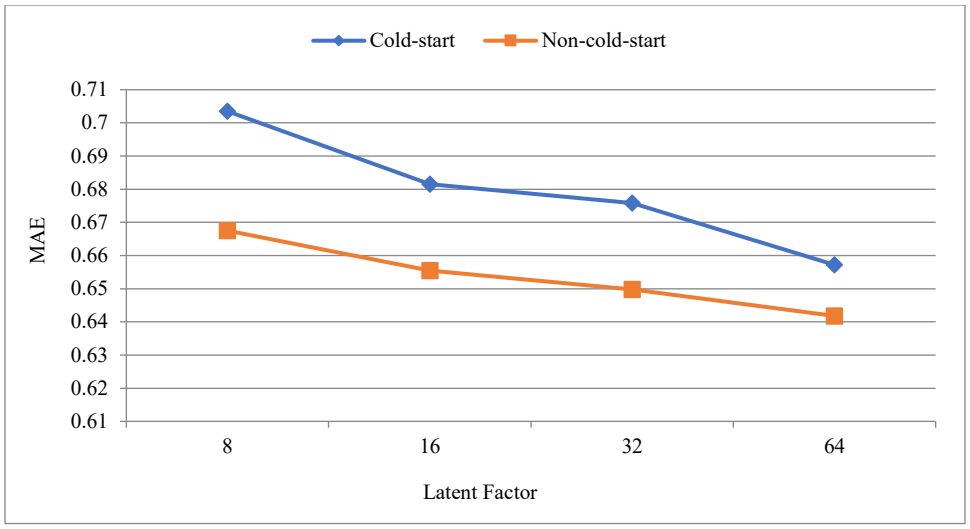


(a)

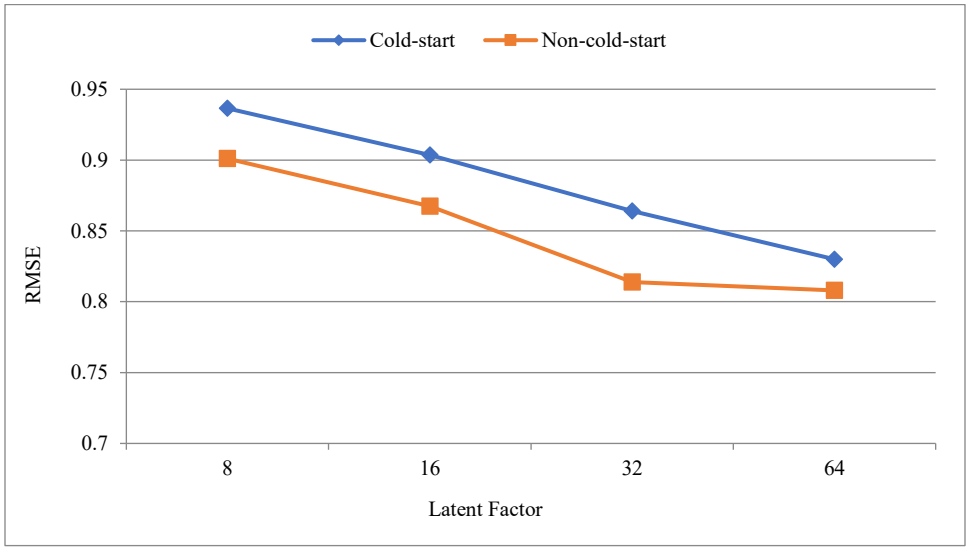


(b)

Fig. 3 Cold-start of conditions prediction on Movie-Lens 100K

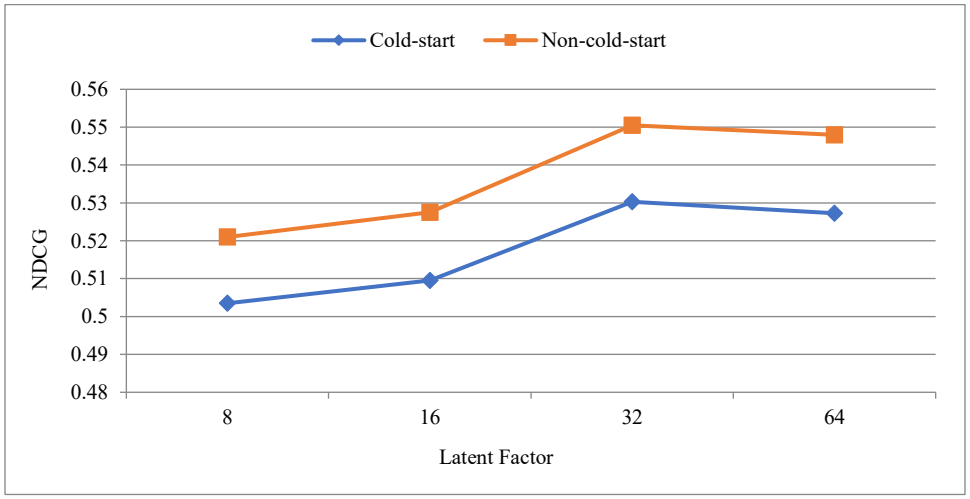


(a)

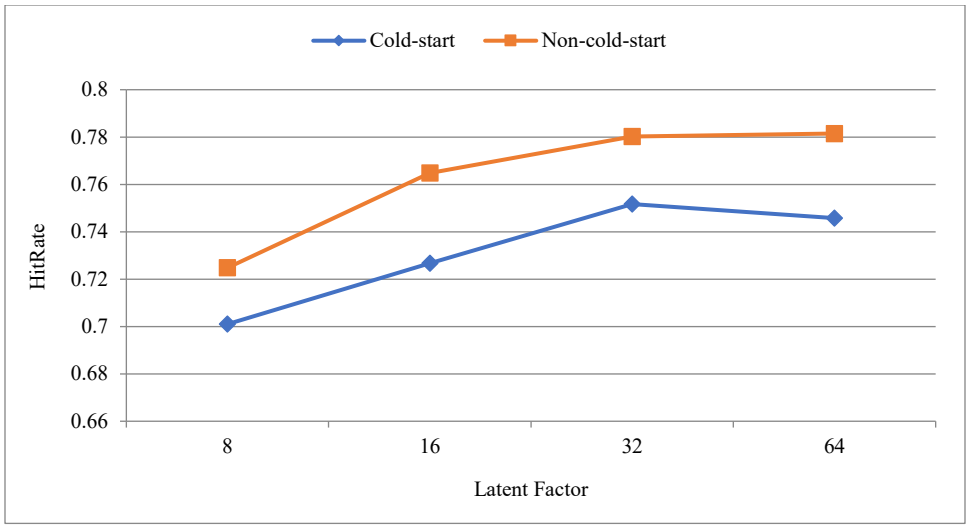


(b)

Fig. 4 Evaluation of cold-start conditions prediction on Movie-Lens 1M

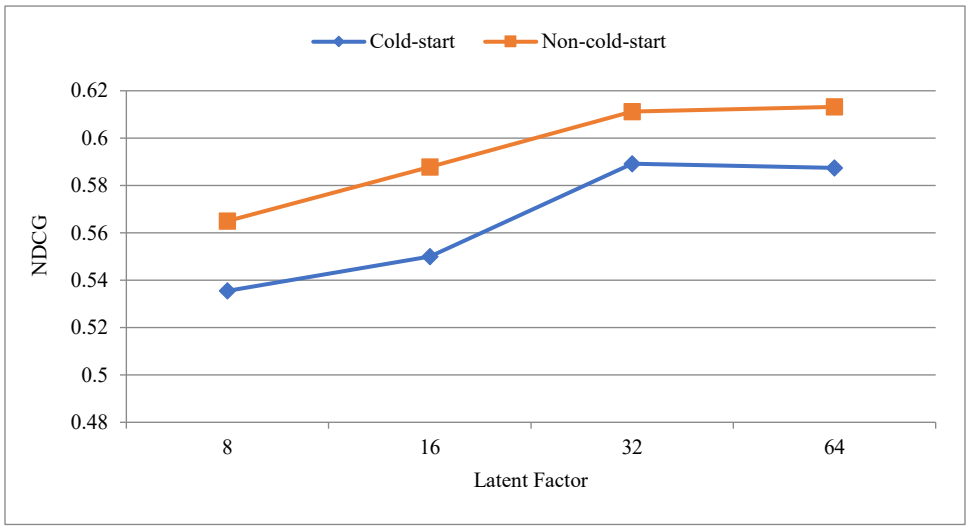


(a)

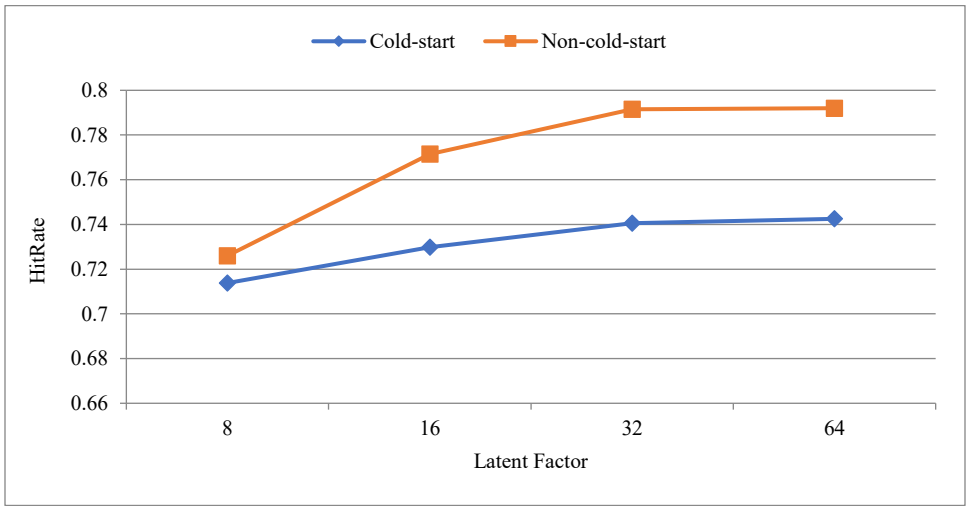


(b)

Fig. 5 Evaluation cold-start conditions ranking on Movie-Lens 100K

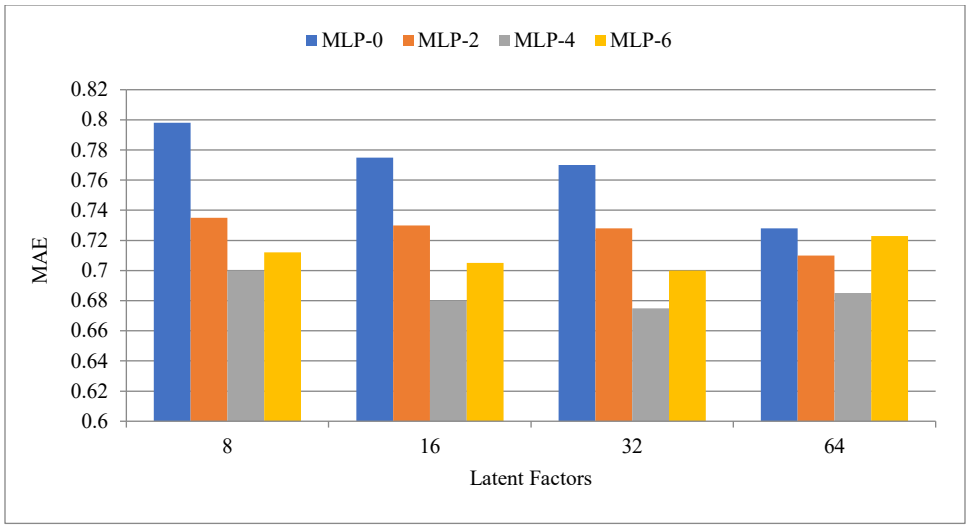


(a)

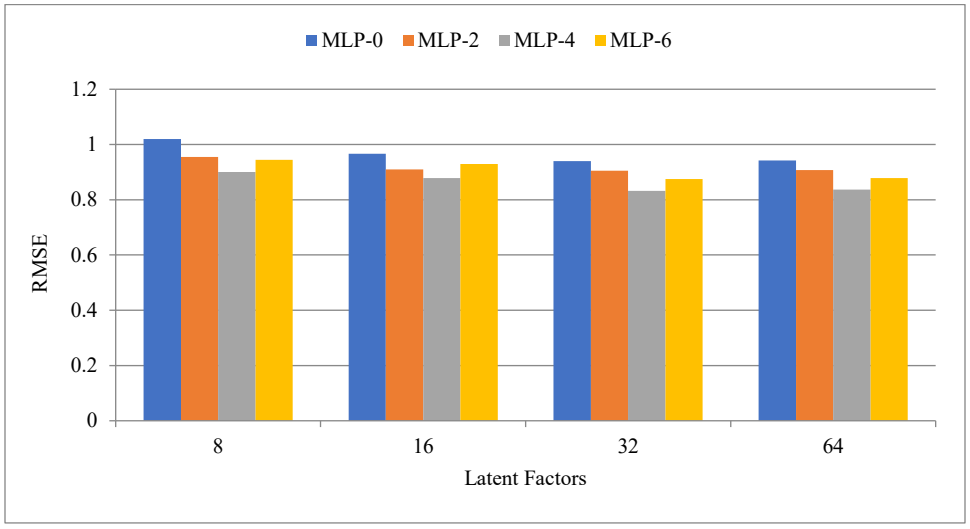


(b)

Fig. 6 Evaluation of cold-start conditions ranking on Movie-Lens

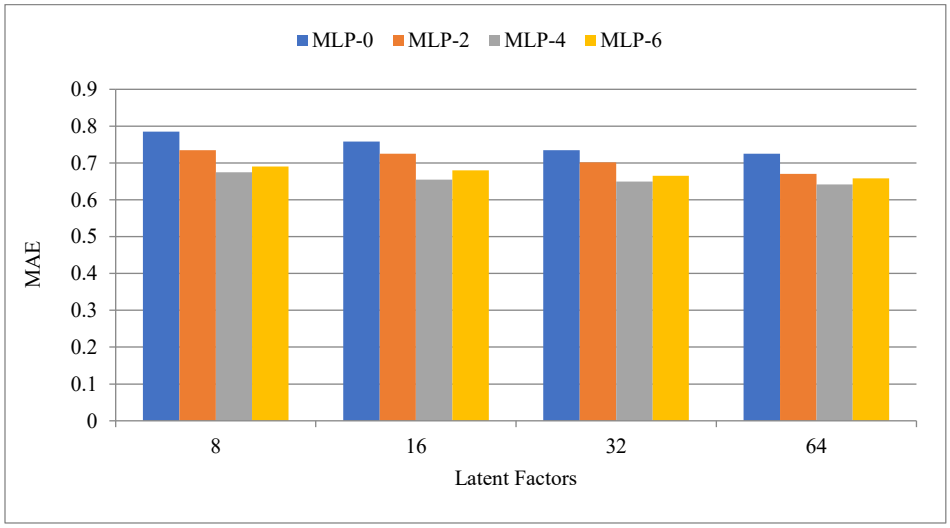


(a)

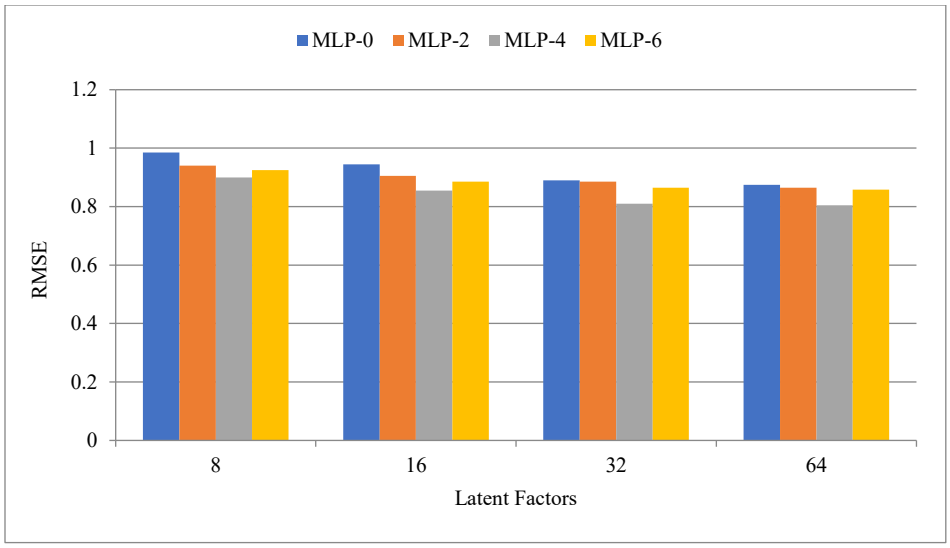


(b)

Fig. 7 Impact of latent factor by varying MLP layer Movie-Lens 100K



(a)



(b)

Fig. 8 Impact of latent factor by varying MLP layer Movie-Lens 100K

5. Conclusion

This paper presented a unified neural recommendation framework for personalized recommendations. The proposed model integrates user and item identity embeddings, additional attribute-based representations, contextual information derived from interaction data, and a learned item-priority feature representation that captures item age, popularity, cold-item status, and item mean rating. By incorporating these components within a single architecture, the proposed model is able to learn richer user-item relationships. Experimental evaluation on the benchmark datasets demonstrated that the proposed model achieves strong performance in both recommendation settings. The results showed that the model provides better rating prediction accuracy in terms of MAE and RMSE. The model is also achieving competitive Top-N recommendation performance in terms of Hit Rate and NDCG. These findings show that the proposed framework is effective not only for accurate score estimation but also for generating ranked recommendation lists. The model was further evaluated under the item cold-start scenario to examine its robustness when newly introduced items have little or no historical interaction information. The results showed only a marginal change in both prediction and ranking performance under this setting, and this shows that newly introduced items were also able to gain visibility in the recommendation process. This indicates that the proposed model does not solely on high-rated or highly interacted items. This highlights the usefulness of the learned item-priority representation in handling sparse-item conditions. For future work, we plan to investigate and apply non-Euclidean representations of side information to model richer and higher-order relationships among users, items, and contextual entities.

Conflicts of Interest

The authors state that no financial, professional, or personal considerations influenced the publication of this work.

Funding Statement

No external funding, grants, or financial assistance was received for conducting or completing this research.

Acknowledgments

The authors appreciate the institutional assistance made available throughout the study. Each author contributed equally to the completion of this research.

References

[1] Jie Lu et al., "Recommender System Application Developments: A Survey," *Decision Support Systems*, vol. 74, pp. 12-32, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[2] Shuaishuai Feng, Junyan Meng, and Jiaxing Zhang, "News Recommendation Systems in the Era of Information Overload," *Journal of Web Engineering*, vol. 20, no. 2, pp. 459-470, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] Francesco Ricci, Lior Rokach, and Bracha Shapira, *Recommender Systems: Techniques, Applications, and Challenges*, Recommender Systems Handbook, pp. 1-35, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[4] Ziyuan Xia et al., "Contemporary Recommendation Systems on Big Data and their Applications: A Survey," *IEEE Access*, vol. 12, pp. 196914-196928, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[5] Gediminas Adomavicius, and Alexander Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[6] Rui Chen et al., "A Survey of Collaborative Filtering-based Recommender Systems: From Traditional Methods to Hybrid Methods based on Social Networks," *IEEE Access*, vol. 6, pp. 64301-64320, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[7] Pasquale Lops et al., "Trends in Content-based Recommendation: Preface to the Special Issue on Recommender Systems based on Rich Item Descriptions," *User Modeling and User-Adapted Interaction*, vol. 29, no. 2, pp. 239-249, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[8] Xiaoyuan Su, and Taghi M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, no. 1, pp. 1-19, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[9] Kawtar Najmani et al., "Collaborative Filtering Approach: A Review of Recent Research," *International Conference on Advanced Intelligent Systems for Sustainable Development*, Rabat, Morocco, pp. 151-163, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Athanasios N. Nikolakopoulos et al., *Trust your Neighbors: A Comprehensive Survey of Neighborhood-based Methods for Recommender Systems*, Recommender Systems Handbook, pp. 39-89, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Badrul Sarwar et al., "Item-based Collaborative Filtering Recommendation Algorithms," *Proceedings of the 10th international conference on World Wide Web*, Association for Computing Machinery, New York, NY, United States, pp. 285-295, 2001. [CrossRef] [Google Scholar] [Publisher Link]

[12] Yehuda Koren, Robert Bell, and Chris Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30-37, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[13] Steffen Rendle, "Factorization Machines," *2010 IEEE International Conference on Data Mining*, Sydney, NSW, Australia, pp. 995-1000, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[14] Steffen Rendle et al., "Neural Collaborative Filtering Vs. Matrix Factorization Revisited," *Proceedings of the 14th ACM Conference on Recommender Systems*, Association for Computing Machinery, New York, NY, United States, pp. 240-248, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Shuai Zhang et al., "Deep Learning based Recommender System: A Survey and New Perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1-38, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[16] Heng-Tze Cheng et al., "Wide and Deep Learning for Recommender Systems," *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, Association for Computing Machinery, New York, NY, United States, pp. 7-10, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[17] Aminu Da'u, and Naomie Salim, "Recommendation System based on Deep Learning Methods: A Systematic Review and New Directions," *Artificial Intelligence Review*, vol. 53, no. 4, pp. 2709-2748, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Hai Chen et al., "Attribute-based Neural Collaborative Filtering," *Expert Systems with Applications*, vol. 185, pp. 1-13, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Hao Wang, Naiyan Wang, and Dit-Yan Yeung, "Collaborative Deep Learning for Recommender Systems," *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Sydney, NSW, Australia, pp. 1235-1244, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[20] Yehuda Koren, "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, New York, NY, United States, pp. 426-434, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[21] Munan Li, Kenji Tei, and Yoshiaki Fukazawa, "An Efficient Adaptive Attention Neural Network for Social Recommendation," *IEEE Access*, vol. 8, pp. 63595-63606, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[22] R. Kiran, Pradeep Kumar, and Bharat Bhasker, "DNNRec: A Novel Deep Learning based Hybrid Recommender System," *Expert Systems with Applications*, vol. 144, pp. 1-14, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[23] Yu Chen, and Sinno Jialin Pan, "Towards Faster Deep Collaborative Filtering Via Hierarchical Decision Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, pp. 4001-4009, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[24] G.M. Harshvardhan et al., "UBMTR: Unsupervised Boltzmann Machine-based Time-Aware Recommendation System," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 6400-6413, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] Ruiqin Wang et al., "Attention-based Dynamic user Modeling and Deep Collaborative Filtering Recommendation," *Expert Systems with Applications*, vol. 188, pp. 1-13, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[26] Shilong Liu et al., "Improving the Performance of Cold-Start Recommendation by Fusion of Attention Network and Meta-Learning," *Electronics*, vol. 12, no. 2, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[27] Wenmin Lin et al., "A Deep Neural Collaborative Filtering based Service Recommendation Method with Multi-Source Data for Smart Cloud-Edge Collaboration Applications," *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 897-910, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[28] Vandana Bhatia, "DLSF: Deep Learning and Semantic Fusion based Recommendation System," *Expert Systems with Applications*, vol. 250, pp. 1-10, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[29] Gopal Behera et al., "Hybrid Collaborative Filtering using Matrix Factorization and Xgboost for Movie Recommendation," *Computer Standards and Interfaces*, vol. 90, pp. 1-8, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[30] Ding Zou et al., "Knowledge Enhanced Multi-Intent Transformer Network for Recommendation," *Companion Proceedings of the ACM Web Conference 2024*, Association for Computing Machinery, New York, NY, United States, pp. 1-9, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Ouahiba Remadnia, Faiz Maazouzi, and Djalel Chefrour, "Hybrid Book Recommendation System using Collaborative Filtering and Embedding based Deep Learning," *Informatica*, vol. 49, no. 8, pp. 189-203, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[32] Guy Aridor et al., "The Movielens Beliefs Dataset: Collecting Pre-Choice Data for Online Recommender Systems," *Proceedings of the 18th ACM Conference on Recommender Systems*, Association for Computing Machinery, New York, NY, United States, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[33] F. Maxwell Harper, and Joseph A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1-19, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[34] Carmel Wenga et al., "A Comprehensive Review on Non-Neural Networks Collaborative Filtering Recommendation Systems," *arXiv preprint*, pp. 1-44, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[35] Devangam Bangaru Rajesh, and Avadhesh Kumar, "Collaborative Filtering Models an Experimental and Detailed Comparative Study," *Scientific Reports*, vol. 15, no. 1, pp. 1-32, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[36] Xiangnan He et al., "Neural Collaborative Filtering," *Proceedings of the 26th International Conference on World Wide Web*, pp. 173-182, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[37] Sophort Siet et al., "Enhancing Sequence Movie Recommendation System using Deep Learning and KMeans," *Applied Sciences*, vol. 14, no. 6, pp. 1-18, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[38] Triyanna Widiyaningtyas et al., "Memory-based Collaborative Filtering based on Matrix Factorization and Gower's Set Rank," *Journal of King Saud University Computer and Information Sciences*, vol. 37, no. 9, pp. 1-35, 2025. [CrossRef] [Google Scholar] [Publisher Link]