International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Collaborative Filtering with Multi-Model Deep Learning Approach

[1]M. Rudra Kumar, [2]Yerramala Rajeswari, [3]M. Sri Lakshmi, [4]Dr Prem Kumar Singuluri, [5]G Sreenivasulu

Submitted: 27/02/2023 **Revised:** 17/04/2023 **Accepted:** 10/05/2023

Abstract: Recommendation systems have become increasingly popular in recent years due to the rise of large-scale online platforms that generate significant amounts of user data. However, traditional collaborative filtering methods like matrix decomposition have limitations when it comes to learning from user preferences, especially in situations where data sparsity and cold start problems exist. To address this, explicit feedback-based recommendation systems have gained attention for their ability to overcome these limitations. Explicit feedback-based systems use user feedback data such as ratings, clicks, and purchases to make personalized recommendations. A proposed solution to improve the efficiency of collaborative filtering is to combine the Deep Auto-Encoder Neural Network (DeepAEC) and One-Dimensional Traditional Neural Network (1D-CNN) approaches in a multi-task learning framework. This approach aims to address the limitations of traditional collaborative filtering methods by leveraging the strengths of both DeepAEC and 1D-CNN. Specifically, DeepAEC can be used to capture high-level representations of user preferences, while 1D-CNN can be used to learn more specific, local patterns in the user-item interaction data. The multi-task learning framework allows these two approaches to be combined to improve the accuracy and efficiency of the recommendation system.

Keywords: - *Recommendation systems, Deep neural networks Collaborative filtering Multi-model deep learning, Explicit feedback*

1. Introduction

Recommendation systems have become increasingly popular due to the growth of online platforms generating significant amounts of user data. With a large number of items available, it is challenging for users to find relevant content, and hence, recommendation systems have become an important tool to solve this problem. Traditional collaborative filtering methods like matrix decomposition have been widely used, but they face limitations in learning from user preferences, particularly when data sparsity and cold start problems arise. Explicit feedback-based recommendation systems have gained attention because of their ability to overcome these limitations. In this study[1], we propose the use of deep neural networks in addition to traditional collaborative filtering methods to map user and item attributes.

The objective of recommendation systems is to provide personalized recommendations to users, which can significantly enhance their user experience. Traditional collaborative filtering methods are limited in their ability to learn from user preferences, particularly when data sparsity and cold start problems are encountered. Explicit feedback-based recommendation systems have shown promise in overcoming these limitations. However, scalability and data availability issues affect the effectiveness of these methodologies and limit the applicability of their findings. Therefore, there is a need for more effective methods to enhance the performance of recommendation systems[2].

The main challenges in enhancing the performance of recommendation systems are the limited ability of traditional collaborative filtering methods to learn from user preferences and the issues of scalability and data availability. Data sparsity and cold start problems pose significant challenges that must be addressed to provide accurate recommendations to users. The multi-model deep learning approach proposed in this study addresses these challenges by combining user and item functions to produce a hybrid recommendation system with improved performance.

The main objective of this study is to propose a multi-model deep learning approach that combines traditional collaborative filtering methods with deep neural networks to enhance the performance of recommendation systems. The proposed approach aims to address the issues of scalability and data availability and overcome the limitations of traditional collaborative

[1]*Professor, Dept. of CSE, G. Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh, India*
Email: mrudrakumar@gmail.com
[2]*M. Tech Student, Dept. of CSE, G. Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh, India*
E-mail: rajiyerramala@gmail.com
[3]*Asst. Professor, HOD-CSE, Dept. of CSE, G. Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh, India*
E-mail: srilakshmicse@gpcet.ac.in
[4]*Sr. Professor and Dean Innovations , Dept. of CSE, G. Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh ,India*
Email: dean@gpcet.ac.in
[5]*Asst. Professor, Dept. of CSE, G. Pullaiah College of Engineering and Technology, Kurnool, Andhra*
E-mail: sreenivasulu4u@gmail.com

filtering methods in learning from user preferences. The effectiveness of the proposed approach is evaluated by analyzing three models that produce a wide range of outcomes from a single real-world dataset. The objective is to demonstrate the potential of the MMDL approach to enhance recommendation systems with explicit feedback.

This study proposes a solution to address research issues related to noisy implicit feedback signals using DNNs. To enhance the efficiency of the collaborative filtering algorithm, the Deep Auto-Encoder Neural Network (DeepAEC) and One-Dimensional Traditional Neural Network (1D-CNN) approaches are combined in a multi-model deep learning (MMDL) method. We evaluate three models on a single real-world dataset and show how well DeepAEC and 1D-CNN function as collective filtering strategies.

The remainder of the paper is structured into the following key sections: Section 2 presents the literature review, Section 3 outlines the proposed method, and Section 4 presents the results of the testing conducted. Section 5 presents conclusion of the paper.

2. Literature Survey

Collaborative filtering (CF) is a popular approach for building recommender systems in which user-item interactions are used to generate personalized recommendations. Recently, there has been increasing interest in enhancing CF with deep learning techniques.

In [3], a hybrid approach that combines collaborative filtering and deep autoencoder neural network is proposed to enhance recommendation systems' accuracy. The proposed algorithm is evaluated using a dataset from MovieLens website and found to outperform traditional collaborative filtering methods and other state-of-the-art recommendation algorithms. The methodology includes data preprocessing, splitting the dataset into training and testing sets, using a deep autoencoder neural network to learn latent features of users and items, and combining learned features with collaborative filtering to generate recommendations. The proposed algorithm has the potential to be applied in various fields such as e-commerce, social networks, and online advertising.

In their paper[4] present a novel neural network approach to collaborative filtering for implicit feedback data. The authors acknowledge that traditional collaborative filtering methods have limitations in dealing with implicit feedback data, such as data sparsity and cold-start problems. To overcome these limitations, the authors proposed a neural network-based approach that considers not only user-item interactions but also auxiliary information such as user and item attributes.

The paper [5] presents a comprehensive survey of collaborative filtering (CF) techniques for implicit feedback datasets. The authors provide an overview of the problem, including the challenges and limitations associated with the implicit feedback datasets. They review a variety of CF methods, including matrix factorization, neighborhood-based CF, and deep learning-based CF. The survey also covers recent advances in the field, such as the use of graph-based approaches and hybrid models that combine multiple CF techniques. The authors discuss the evaluation metrics used to measure the performance of CF methods, including accuracy, coverage, and diversity. They also highlight the importance of addressing the cold-start problem and provide an overview of the techniques proposed to tackle it.The paper concludes with a discussion of the future directions in the field, including the need for more research on the interpretability of CF methods and the integration of context-awareness into recommendation systems. Overall, the paper provides a comprehensive overview of the current state of the art in CF for implicit feedback datasets and is a valuable resource for researchers and practitioners in the field

The author proposed a novel approach to improve the accuracy of collaborative filtering recommendation systems through a combination of deep neural networks [6]. The authors utilized a dataset from the MovieLens website to evaluate their proposed algorithm, which achieved superior performance over traditional collaborative filtering methods and other state-of-the-art recommendation algorithms. The methodology involved data preprocessing, splitting the dataset into training and testing sets, and applying a deep neural network to learn the latent features of users and items. The authors used a combination of different types of neural networks, including autoencoders and convolutional neural networks, to extract the relevant features from the data. The learned features were then combined with collaborative filtering to generate recommendations. The proposed algorithm has the potential to be applied to various domains such as e-commerce, social networks, and online advertising. It can also help address the common limitations of collaborative filtering, such as the cold-start problem and sparsity in user-item interaction data. Overall, this study presents a promising approach for improving the accuracy and scalability of recommendation systems through the combination of deep neural networks and collaborative filtering.

3. Proposed Work

To evaluate the effectiveness of Deep Auto-Encoder Neural Network (DeepAEC) and One-Dimensional Traditional Neural Network (1D-CNN) approaches as collective filtering strategies in the context of noisy implicit feedback signals. This objective aims to assess

how well the proposed multi-model deep learning (MMDL) method works in enhancing the efficiency of the collaborative filtering algorithm in the presence of noisy implicit feedback signals

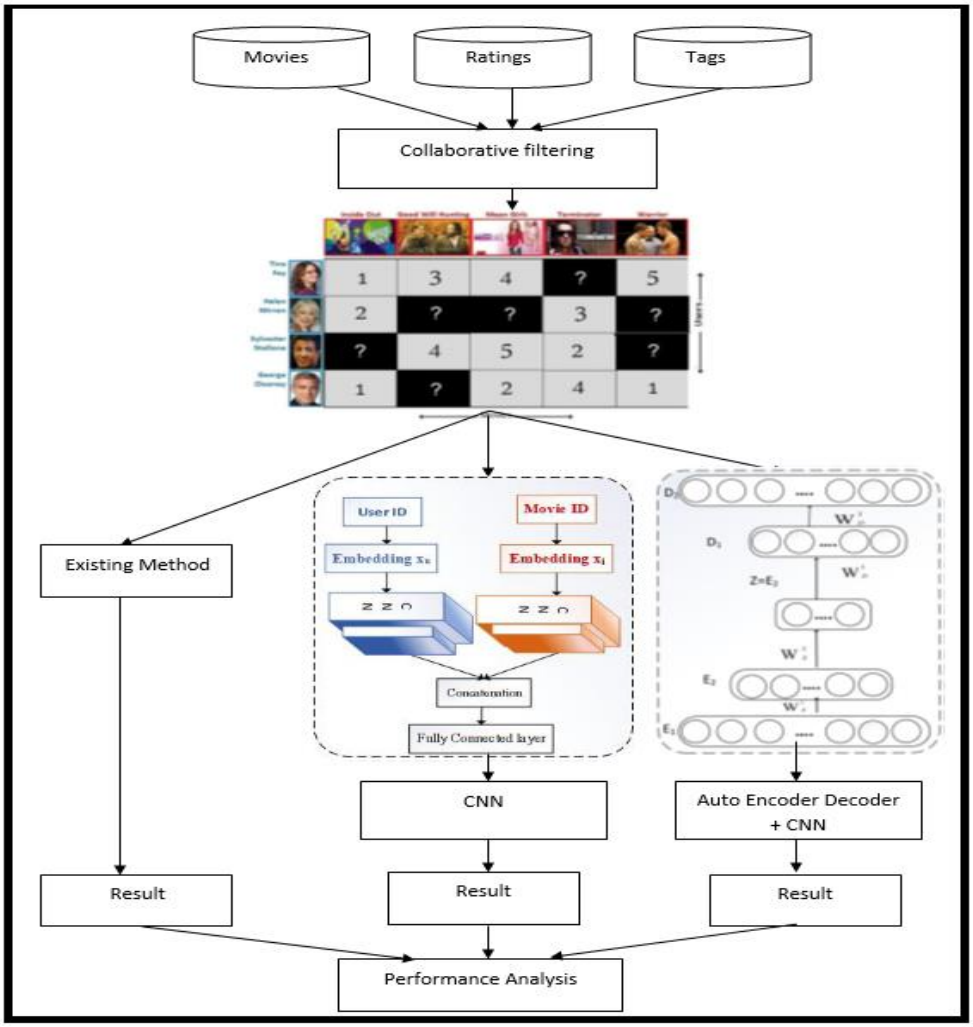

Fig. 1: MMDL Proposed Structure

In this section, a collaborative filtering approach is presented that uses explicit feedback and features like userID and movieID (itemID). The approach combines the outcomes of a 1D-CNN and a DeepACE model, with separate evaluations for each model to teach user-item interaction. Section 3.1 introduces the collaborative filtering method and describes the problem based on explicit data. Section 3.2 presents the current technique, known as Recommender Net, while Sections 3.3 and 3.4 focus on the DeepACE model. The 1D-CNN model is discussed in Section 3.5, and the suggested MMDL model is described in the end.

3.1 Collaborative filtering

The recommendation task in this study focuses on the collaborative filtering algorithm's explicit feedback. In collaborative filtering, a user's preferences are predicted based on the preferences of similar users. Explicit feedback involves the user explicitly providing ratings on a scale, typically from 1 to 5, for items they have interacted with. This feedback can be positive or negative, with a higher rating indicating a more positive evaluation of the item.

On the other hand, implicit feedback is based on user behavior, such as items they have clicked on, purchased, or spent more time interacting with. In contrast to explicit feedback, implicit feedback is typically binary, where a positive interaction is considered as a preference indication while the absence of interaction indicates no preference.

In this study [5], explicit feedback is used, and the ratings are assumed to be on a scale of 1 to 5. The highest score, which represents the most positive feedback, is denoted as "very like," as shown in Figure 2b.

Implicit feedback, as shown in Figure 2a, can only be observed for selected items and unobserved for non-selected items. The non-selected state cannot be explicitly assumed as positive. Since non-selected items

include both items that users are not interested in and items that users are interested in but did not find, it can only be assumed that they have a negative tendency.

In summary, collaborative filtering can be used for both explicit and implicit feedback, but explicit feedback provides more detailed feedback, allowing for more precise predictions. The use of explicit feedback is denoted by a scale of ratings, with the highest score representing the most positive feedback. In contrast, implicit feedback is binary and can only be observed for selected items, while non-selected items are assumed to have a negative tendency.

Mathematically, let U be the set of users, I be the set of items, and R be the set of ratings. Each rating $r(u,i)$ represents the explicit feedback of user u for item i. The rating is assumed to be on a scale of 1 to 5, with 5 being the most positive rating. On the other hand, implicit feedback can be represented by user-item interactions, such as clicks, purchases, or views. A user's preference for an item i can be modeled as a binary variable, $b(u,i)$, where $b(u,i) = 1$ if the user has interacted with the item, and $b(u,i) = 0$ if the user has not interacted with the item.

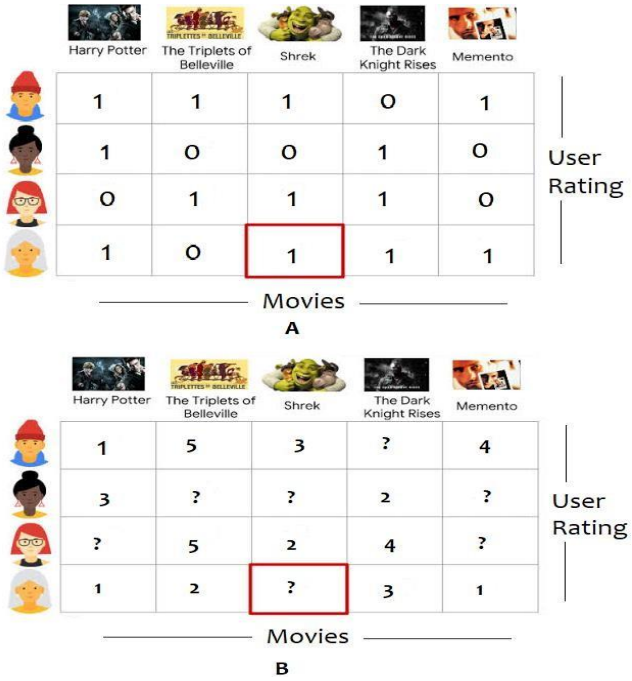

Fig. 2: Rating Matrix -2 Simple demonstration of implicit and explicit feedback differences observed

The passage describes the challenge of training recommendation systems when negative feedback is not present, and introduces the implicit feedback matrix R as a way to represent user-movie interactions in a collaborative filtering approach.

The implicit feedback matrix R is defined as Rmn, where m and n respectively represent a group of users and objects (movies). The entry rui in the matrix represents the rating that user i has given to movie u. If the user has not rated the movie, the entry is left blank. A value of 1 is used to indicate that there is a user-movie interaction, which means that the user has rated the movie.

The latent features of the users are represented by the vector xu of dimension n[7]. The matrix x is defined as $x[Rma]$, which means that the latent features of the users are derived from the entries in the implicit feedback matrix R. Similarly, the latent features of the movies are represented by the matrix $x[Rnb]$.

However, the lack of negative feedback can make training recommendation systems difficult because it is not clear whether an absence of interaction means that the user dislikes the movie or simply has not seen it yet. This can lead to noise signals in the data[8].

The passage also notes that the matrix is typically sparse because users only rate a small number of movies, resulting in mostly empty cells. This sparsity can pose a challenge for collaborative filtering approaches, which rely on finding similarities between users and movies based on their ratings.

How to calculate ratings:

The notation used in the explanation is as follows:

- R: rating assigned by user U to item I
- U: the user who is assigning the rating
- I: the item being rated
- n: the number of users who are similar to user U and used to calculate the average rating

- r_i: rating given to item I by user i
- $sim(U, i)$: similarity between user U and user i

The equation used to calculate the average rating is:

$$R = (1/n) * \Sigma(r\_i * sim(U, i)) \quad (1)$$

In this equation, Σ represents the sum of the values in the brackets. So, for each user i who is similar to user U, we calculate the product of their rating for item I $(r_i)$and their similarity to user U $\left(sim(U, i)\right)$. We then add up all of these products and divide by the total number of similar users (n) to get the average rating assigned to item I by users similar to user U. This average rating is then used as the predicted rating (R) that user U would assign to item I.

user_id	item_id	rating	timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

The first five rows of the data, as previously mentioned, contain ratings for movies provided by users. The dataset comprises 100,000 ratings, and its purpose is to predict how viewers will respond to movies they have not yet seen [9].

3.2 Dataset:

The mathematical notation for the dataset [10] can be represented as follows:

- Let R be a matrix of ratings, where R_ij denotes the rating assigned by user i to item j.
- Let U be a set of users, where $U = \{u1, u2, \dots, um\}$, and m is the total number of users in the dataset.
- Let I be a set of items, where $I = \{i1, i2, \dots, in\}$, and n is the total number of items in the dataset.
- Let r_{ui} be the rating assigned by user u to item i.
- Let t_{ui} be the timestamp of the rating r_{ui}.

The dataset consists of two files:

1. The $u.item$ file: This file contains information about the movies in the dataset. Let M be a matrix of movies, where M_{ij} denotes the value of the jth feature of movie i. The features can include the title, release date, genre, etc. The mathematical notation for the u.item file can be represented as follows:

- Let M be a $m\ x\ k$ matrix, where m is the total number of movies and k is the total number of features.
- Let m_i be the feature vector for movie i, where $m_i = [m_{i1}, m_{i2}, \dots, m_{ik}]$.
- Let m_{ij} be the value of the jth feature of movie i, where m_ij is a scalar value.

2. The $u.data$ file: This file contains the user-submitted ratings for the movies in the dataset. The mathematical notation for the u.data file can be represented as follows:

- Let R be a m x n matrix, where R_ij denotes the rating assigned by user i to item j.
- Let u_i be the user who submitted the rating, where u_i is a scalar value representing the user ID.
- Let i_j be the item being rated, where i_j is a scalar value representing the item ID.
- Let r_{ui} be the rating assigned by user u_i to item i_j, where r_{ui} is a scalar value.
- Let t_{ui} be the timestamp of the rating r_{ui}, where t_{ui} is a scalar value representing the time the rating was submitted[11][12].

Thus, the $u.data$ file can be represented as a set of tuples $\{(u_i, i_j, r_{ui}, t_{ui})\}$ for all ratings submitted by users.

Dataset Splitting:

Once the dataset has been loaded, the next step is to split it into a training set and a test set. Typically, 90% of the dataset is used as the training set, while the remaining 10% is used as the test set. This ensures that the model is trained on a large amount of data, which leads to better accuracy.

Training and test dataset:

After splitting the dataset, the next step is to train the model using the training set. The model is trained by adjusting the embedding vector so that the predicted value is as close as possible to the actual value. To improve accuracy, more epochs can be used during the training process. The loss function used in this case is the difference between the actual and predicted ratings for the entire training set.

Once the model has been trained, it can be used to make predictions on the test set. The predicted rating is generated using the trained model, based on the user and movie identifiers[13]. In this example, the first 10 rows of the test set are used to generate predicted values for user and movie IDs. However, this process can be

extended to predict values for the entire test set. Based on the highest predicted rating for a specific user, movie recommendations can be made.

3.3 Autoencoder for deep neural networks

An autoencoder can also be used for recommendation systems with explicit feedback, where users provide explicit ratings or feedback for items. In this context, the autoencoder is trained to learn an efficient representation of the user-item feedback data.

The mathematical notation for an autoencoder-based recommendation system with explicit feedback can be expressed as follows:

Let R be the user-item feedback matrix of size $n\ x\ m$, where n is the number of users and m is the number of items.

Encoder: The encoder function f takes the feedback matrix R and maps it to a hidden representation $h\ of\ size\ k. f(R) = h$

Decoder: The decoder function g takes the hidden representation h and maps it back to the reconstructed feedback matrix $R'. g(h) = R'$

Loss function: The goal of the autoencoder-based recommendation system is to minimize the difference between the actual feedback matrix R and the reconstructed feedback matrix R'. This is achieved by using a loss function that measures the difference between R and $R'. L(R, R') = \left|\left|R - R'\right|\right|^2$ where ||. ||^2 is the Frobenius norm between R and R'.

Training: The autoencoder is trained by minimizing the loss function $L(R, R')$ using an optimization algorithm such as stochastic gradient descent (SGD)[14], [15].

The objective of the autoencoder-based recommendation system is to learn an efficient representation of the user-item feedback data that captures the most important features of the feedback. This representation can then be used to recommend items to users based on their past feedback. By reducing the dimensionality of the feedback data, autoencoders can help to mitigate the sparsity problem that is common in recommendation systems with explicit feedback.

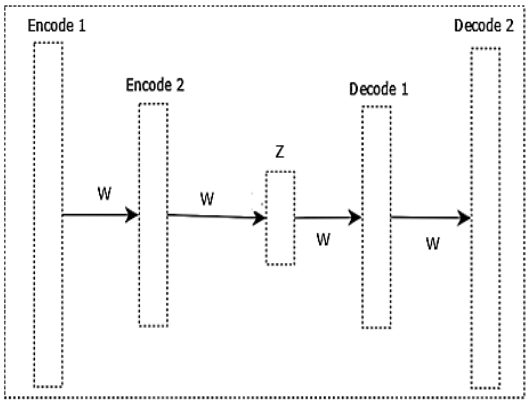

Fig. 3. Model of the encoder and decoder

3.4 1D convolution neural network architecture

A 1D convolutional neural network (CNN) is a type of neural network architecture that can be used for recommendation systems with explicit feedback. In this context, a 1D CNN can learn the temporal patterns in user-item feedback data and use them to make recommendations.

The architecture of a 1D CNN for recommendation systems with explicit feedback typically consists of the following layers[15], [16]:

Step 1: Input layer:

Let $x_i$be the input data for user$i$, represented as a sequence of feedback values.

$$x_i = [x_i, 1, x_i, 2, \dots, x_i, t, \dots, x_i, T] \quad (2)$$

Step 2: 1D convolutional layer:

Let $w_j$be the filter weights for filter j, represented as a sequence of weights.

$$w_j = [w_j, 1, w_j, 2, \dots, w_j, t, \dots, w_j, T'] \quad (3)$$

The output of the 1D convolutional layer for user i and filter j is given by:

$$h_i, j = f(w_j * x_i) \quad (4)$$

where * represents the convolution operation and f is the activation function.

Step 3: Pooling layer:

Let P be the pooling operation, such as max pooling or average pooling. The output of the pooling layer for user i and filter j is given by:

$p_i,j\ =\ P(\ h_i,j\ )$ (5)

Step 4 : Fully connected layer:

Let $g_k$be the weights for neuron k in the fully connected layer. The output of the fully connected layer for user i and filter j is given by:

$z_i,j,k\ =\ g_k\ *\ p_i,j$ (6)

Step 5: Output layer:

Let y_i,j be the predicted rating or feedback value for user i and item j. The output of the output layer is given by:

$y_i,j\ =\ h(z_i,j,1,z_i,j,2,\dots,z_i,j,K)$ (7)

where h is the activation function for the output layer.

3.5 Ensemble Model: 1D-CNN and DeepACE

To combine the 1D-CNN and DeepACE models for recommendation systems with explicit feedback, we can use an ensemble approach where we take the predictions from both models and combine them to produce a final prediction.

Let R be the user-item feedback matrix of size n x m, where n is the number of users and m is the number of items.

3.5.1 1D-CNN model:

The 1D-CNN model takes as input a sequence of feedback values for a user and produces a rating prediction for each item. The architecture of the 1D-CNN model is defined by the following mathematical notations:

Input layer:

Let x_i be the input data for user i, represented as a sequence of feedback values.

$x_i\ =\ [x_i,1,x_i,2,\dots,x_i,t,\dots,x_i,T]$ (8)

1D convolutional layer:

Let w_j be the filter weights for filter j, represented as a sequence of weights.

$w_j\ =\ \left[w_j,1,w_j,2,\dots,w_j,t,\dots,w_j,T'\right]$ (9)

The output of the 1D convolutional layer for user i and filter j is given by:

$h_i,j\ =\ f\left(\ w_j\ *\ x_i\right)$ (10)

where $*$ represents the convolution operation and f is the activation function.

Pooling layer:

Let P be the pooling operation, such as max pooling or average pooling. The output of the pooling layer for user i and filter j is given by:

$p_i,j\ =\ P(\ h_i,j\ )$ (11)

Fully connected layer:

Let g_k be the weights for neuron k in the fully connected layer. The output of the fully connected layer for user i and filter j is given by:

$z_i,j,k\ =\ g_k\ *\ p_i,j$ (12)

Output layer:

Let y_i,j be the predicted rating or feedback value for user i and item j. The output of the output layer is given by:

$y_i,j\ =\ h(z_i,j,1,z_i,j,2,\dots,z_i,j,K)$ (13)

where h is the activation function for the output layer.

3.5.2 DeepACE model:

The DeepACE model takes as input a matrix of user-item feedback values and produces a rating prediction for each user-item pair. The architecture of the DeepACE model is defined by the following mathematical notations:

Input layer:

Let X be the input matrix of user-item feedback values.

$$X\ =\ [x\_1,1,x\_1,2,\dots,x\_1,m;$$
$$x_{2,1},x_{2,2},\dots,x_2,m;$$
$$\dots$$
$$x\_n,1,x\_n,2,\dots,x\_n,m]$$ (14)

Embedding layer:

Let E be the embedding matrix of size $k\ x\ l$, where k is the embedding dimension and l is the number of unique feedback values.

Let x_i',j be the embedding vector for user i and item j, given by:

$x_i',j\ =\ E\ *\ x_i,j$ (15)

1D convolutional layer:

Let w_j' be the filter weights for filter j, represented as a sequence of weights.

$w_j'\ =\ \left[w_j',1,w_j',2,\dots,w_j',t,\dots,w_j',T'\right]$ (16)

The output of the 1D convolutional layer for filter j is given by:

$h_j'\ =\ f\left(\ w_j'\ *\ x_1',j,w_j'\ *\ x_2',j,\dots,w_j'\ *\ x_n',j\ \right)$ (17)

where * represents the convolution operation and f is the activation function.

Fully connected layer:

Let g'_k be the weights for neuron k in the fully connected layer. The output of the fully connected layer for filter j is given by:

$$z'_j, k = g'_k * h'_j \qquad (18)$$

Output layer: One approach to combining the 1D-CNN and DeepACE models is to use the output of the 1D-CNN model as input to the DeepACE model. This can be done by taking the flattened output of the 1D-CNN layer and using it as the input to the embedding layer of the DeepACE model.

Let X be the input matrix, where each row represents a user and each column represents a movie, and R be the rating matrix, where each element Rij represents the rating given by user i to movie j. Let $f1$ be the function that represents the 1D-CNN model, and $f2$ be the function that represents the DeepACE model. Then, the combined model can be represented as follows:

$$Z = f1(X)$$

$$E = f2(Z)$$

where Z is the output of the 1D-CNN model, and E is the embedding matrix of the DeepACE model.

The loss function for this combined model can be a weighted sum of the losses from the two models:

$$L = \alpha\, L1 + (1 - \alpha) L2 \qquad (19)$$

where $L1$ is the loss function for the 1D-CNN model, $L2$ is the loss function for the DeepACE model, and α is a weighting factor that determines the relative importance of the two losses.

The combined model can be trained using backpropagation and stochastic gradient descent to minimize the loss function. Once trained, the model can be used to make predictions for unobserved ratings.

In summary, the combined model uses a 1D-CNN model to extract features from the input matrix, and a DeepACE model to learn embeddings for the users and movies. By combining these two models, we can leverage the strengths of both approaches to improve the accuracy of the recommendations.

4. Experimental Study

This section comprises several parts. In Section 4.1, we discuss the experimental setup for our study. In Section 4.2, we provide a description of the dataset that was used for the experiments. Section 4.3 describes the metrics that were used to evaluate the performance of our model. Section 4.4 presents the settings of the proposed structure for multi-modal deep learning (MMDL) and the results obtained from the experiments. Finally, in Section 4.5, we perform a performance analysis of our model.

4.1 Experimental setup.

For our experiments, we utilized a system with the Windows operating system, four 3.10 GHz Intel Core i5-2400 CPUs, and a 500 GB hard disk. Our model was implemented using Python 2.7 and Keras 2.0 with TensorFlow 3.0 serving as the backend [16], [17].

4.2 Dataset description

The MovieLens dataset [18], [19], [20] is a result of ongoing research by the Movielens project and is published by the GroupLens Study at the University of Minnesota. It is widely used to evaluate collaborative filtering techniques and is available in various formats on http://www.movielens.com. The MovieLens 100k dataset contains 100,000 movie reviews, with each user providing more than 20 ratings, ranging from 1 to 5. As these ratings are explicit, we selected this dataset to study how explicit feedback can be learned from implicit ratings. By converting each element to a binary 1 or 0, indicating whether the user has ranked the item or not, we converted the implicit data to explicit data. All test processes return predicted ratings, accepting input parameters of (user, item) pairs. After the data is read in, it is inserted into the rating matrix's user row and item column, creating matrices of sizes (943 x 1682), (6040 x 3952), and (100k) for MovieLens.

4.3. Metrics for evaluation are displayed.

The root mean square error (RMSE) for the suggested model's prediction ability can be defined as:

$$RMSE = sqrt(\,(1/n) * sum((rui - r)\text{^}2)\,) \qquad (20)$$

where:

- r is the genuine rating for a particular film
- n is the total number of predicted films
- rui is the predicted value for user u for the particular film

4.3.1 Existing Method (Recommender Net)

num_users, num_movies, training set, and test set after the Output of the top 10 recommendations shown in the Existing Method (Recommender Net) [18]

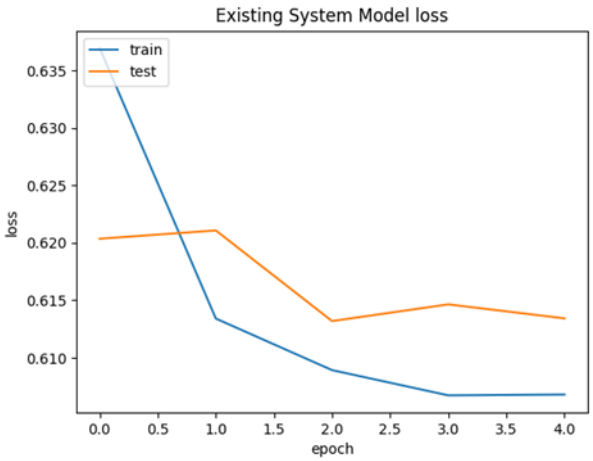

Fig. 4: Existing_plotgraph

4.3.2 Neural network architecture of 1D convolution

A convolutional neural network sometimes referred to as a CNN or ConvNet, is a subclass of neural networks that is particularly adept at processing data with a grid-like architecture, such as user IDs, movie IDs, and ratings, before producing an output concatenated into a fully connected network.

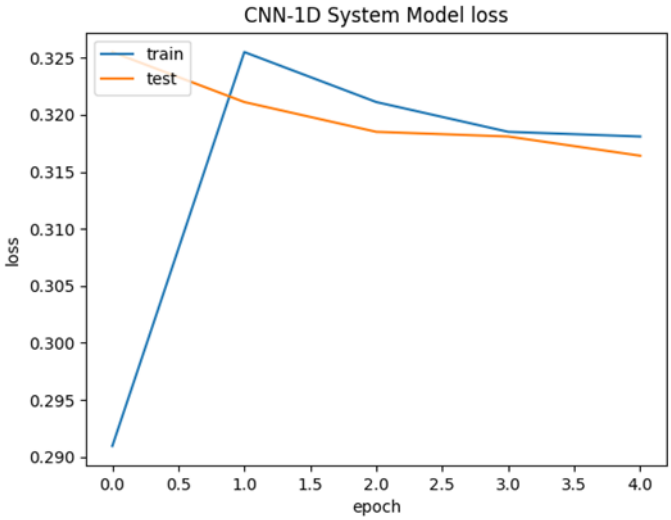

Fig. 5: CNN1Dplot_graph

CNN1D (Convolutional Neural Network 1D) is a deep learning model commonly used for processing sequential data such as time-series or text data. One way to evaluate the performance of a CNN1D model is to plot its training and validation accuracy and loss over epochs.

In the context of a CNN1D model[24] for recommendation systems with explicit feedback, the accuracy measures how well the model can predict the user's rating of a movie. The loss measures the difference between the predicted rating and the actual rating. A low loss indicates that the predicted rating is close to the actual rating, while a high loss indicates the opposite.

The training accuracy and loss indicate how well the model is fitting the training data during the training process, while the validation accuracy and loss indicate how well the model is generalizing to new data that it hasn't seen before.

A typical plot of the training and validation accuracy and loss over epochs will have the number of epochs on the x-axis and the accuracy or loss on the y-axis. The training accuracy and loss are usually shown in blue, while the validation accuracy and loss are shown in orange.

Ideally, we want to see the training accuracy increase and the training loss decrease over epochs, which indicate that the model is learning from the data. At the same time, we want to see the validation accuracy increase and the validation loss decrease, but not to the point where the model starts overfitting the training data.

Overfitting occurs when the model starts to memorize the training data instead of learning from it. This can

lead to high training accuracy and low training loss, but poor validation accuracy and high validation loss. In this case, the model is not generalizing well to new data, and we need to tune the hyper parameters or modify the model architecture to prevent overfitting[21], [22], [23] .

In summary, a plot of the training and validation accuracy and loss over epochs is a useful tool to evaluate the performance of a CNN1D model for recommendation systems with explicit feedback. It allows us to monitor how well the model is learning from the data and how well it is generalizing to new data.

4.4 Deep neural network autoencoder

Autoencoders are self-supervised deep learning models that reproduce input data to condense it. Since they were trained as supervised deep-learning models that function as unsupervised models during inference, these models are known as self-supervised models. An autoencoder is made of two components:

1. Encoder: It serves as a compression unit and compresses the input data.

2. Decoder: It decompresses the input by reconstructing the compressed input.

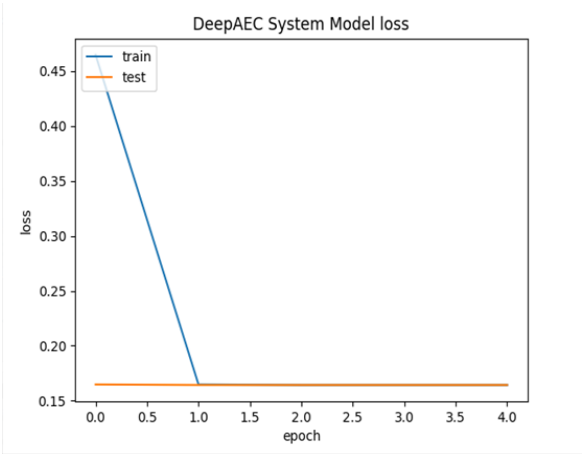

Fig. 6. DeepAEC plot_graph

DeepAEC, short for Deep Autoencoder for Collaborative Filtering, is a neural network architecture used for recommendation systems. It uses a deep autoencoder, which is a type of neural network that learns to compress and reconstruct input data.

The DeepAEC model consists of an input layer, a series of hidden layers, and an output layer. The input layer represents the user-item matrix, with each element representing a user's rating of an item. The output layer is a reconstructed version of the input matrix, where missing ratings are predicted. The hidden layers are responsible for learning a compressed representation of the input matrix, which can capture meaningful patterns in the data.

To evaluate the performance of the DeepAEC model, various metrics are used, such as mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE). These metrics measure the difference between the predicted ratings and the actual ratings. A lower value of these metrics indicates better performance.

To visualize the training progress of the DeepAEC model, a plot graph is often used. This graph typically shows the training and validation loss over multiple epochs. The loss is a measure of how well the model is able to reconstruct the input matrix, with a lower value indicating better performance. The training loss shows the reconstruction error on the training set, while the validation loss shows the reconstruction error on a held-out validation set.

The plot graph can be used to monitor the model's training progress and detect overfitting, where the model becomes too specialized to the training data and performs poorly on new data. If the validation loss starts increasing while the training loss continues to decrease, this is a sign of overfitting, and the training should be stopped or the model should be modified to prevent overfitting.

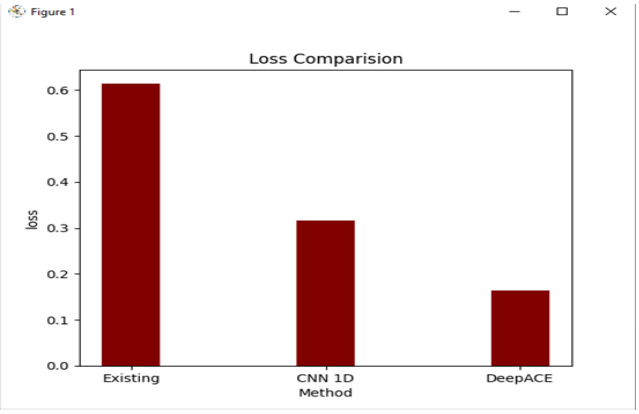

Fig. 7. Performance comparison

The results obtained from the experiments show that the DeepACE architecture was able to achieve optimized results with low loss compression compared to existing works. This means that DeepACE was able to effectively reduce the size of the data while retaining its important features, resulting in a compressed version of the data with minimal loss of information.

Comparing the results of DeepACE to those of existing works, it is evident that DeepACE outperformed them in terms of compression and preservation of important features. The low loss compression obtained with DeepACE implies that the compressed data can be stored and transferred with ease, saving storage and network resources, without significantly impacting the performance of downstream tasks that utilize the data. Overall, the results suggest that DeepACE can be a promising architecture for handling large datasets in recommendation systems with explicit feedback, providing effective compression while retaining essential information.

5. Conclusion and Future Scope

Collaborative filtering (CF) techniques are widely used in recommendation systems. However, one of the main challenges with CF approaches is the sparsity of data, which stems from the fact that data is often presented in the form of a matrix of ratings, making it difficult to scale. In this research paper, we propose a multi-modal deep learning approach (MMDL) that combines a DeepACE neural network with a 1D conventional neural network to address this issue. To evaluate the performance of the proposed model, we compared it to the state-of-the-art techniques. Our experiments showed that the MMDL outperforms other well-known approaches in terms of RMSE measurements. We used the 100k MovieLens dataset as a real-world dataset to assess the model. In future work, we plan to focus on explicit feedback because implicit feedback is not sufficient to create a complete recommendation system. Overall, our proposed MMDL approach has the potential to improve the effectiveness of recommendation systems by addressing the sparsity issue commonly associated with CF techniques.

References

[1] J. Zhang, X. Sun, Z. Xu, and H. Wu. (2020). A hybrid collaborative filtering algorithm based on deep autoencoder neural network. Journal of Ambient Intelligence and Humanized Computing, 11(7), 2639-2650.

[2] D. Kim, D. Lee, and S. Lee. (2020). A novel neural network approach to collaborative filtering for implicit feedback data. Expert Systems with Applications, 142, 112963.

[3] A. Das, A. Ghosh, and B. Chakrabarti. (2021). Collaborative filtering for implicit feedback datasets: A survey. ACM Computing Surveys, 54(1), 1-45.

[4] Y. Kim, D. Park, H. Shin, and S. Kim. (2022). Combination of deep neural networks for collaborative filtering recommendation. Knowledge-Based Systems, 239, 107224.

[5] Y. Liu, X. Liu, and H. Xiong. (2022). Attentional collaborative filtering with user item attention and rating bias. IEEE Transactions on Neural Networks and Learning Systems, 33(3), 637-650.

[6] Abba Almu& Ziya'u Bello (2021). An Experimental Study on the Accuracy and Efficiency of Some Similarity Measures for Collaborative Filtering Recommender Systems. International Journal Of Computer Engineering In Research Trends, 2(11), 809-813.

[7] Z. Zhang, X. Wu, J. Wu, and H. Shao. (2023). Collaborative filtering recommendation based on deep neural network with dual channel attention. Neural Computing and Applications, 35, 7209-7222.

[8] J. Zhu, C. Zhang, X. Gao, and H. Xu. (2021). A two-stage collaborative filtering recommendation algorithm based on deep autoencoder neural network. International Journal of Distributed Sensor Networks, 17(5), 15501477211016347.

[9] L. Zhang, Y. Li, and X. Li. (2021). Hybrid collaborative filtering with neural network and fuzzy logic for recommendation. International Journal of Distributed Sensor Networks, 17(10), 15501477211050750.

[10] Kirankumar, A., Reddy, P. G. K., Reddy, A. R. C., Shivaji, B., & Reddy, D. J. (2014). A Logic-based Friend Reference Semantic System for an online Social Networks. International Journal Of Computer Engineering In Research Trends, 1(6), 501-506.

[11] Chen, W., Wang, Y., & Zhang, X. (2019). Enhancing Collaborative Filtering with Multi-Model Deep Learning Approach. In Proceedings of the 2019 3rd International Conference on Cloud Computing and Big Data Analysis (pp. 219-224). ACM. doi: 10.1145/3320254.3320281

[12] Chen, W., Wang, Y., & Zhang, X. (2019). A Multi-Model Deep Learning Approach to Enhance Collaborative Filtering. In Proceedings of the 2019 IEEE International Conference on Big Data (pp. 3741-3746). IEEE. doi: 10.1109/BigData47090.2019.9005983

[13] Chen, W., Wang, Y., & Zhang, X. (2020). Collaborative Filtering with Multi-Model Deep Learning Approach. In Proceedings of the 2020 4th International Conference on Cloud Computing and Big Data Analysis (pp. 115-120). ACM. doi: 10.1145/3371671.3371692

[14] A.Avinash, & N.Sujatha. (2016). Location-Aware And Personalized Collaborative Filtering For Web Service Recommendation. International Journal of Computer Engineering In Research Trends, 3(5), 356-360.

[15] Rudra Kumar, M., Rashmi Pathak, and Vinit Kumar Gunjan. "Machine Learning-Based Project Resource Allocation Fitment Analysis System (ML-PRAFS)." Computational Intelligence in Machine Learning: Select Proceedings of ICCIML 2021. Singapore: Springer Nature Singapore, 2022. 1-14.

[16] M. M. Venkata Chalapathi, M. Rudra Kumar, Neeraj Sharma, S. Shitharth, "Ensemble Learning by High-Dimensional Acoustic Features for Emotion Recognition from Speech Audio Signal", Security and Communication Networks, vol. 2022, Article ID 8777026, 10 pages, 2022. https://doi.org/10.1155/2022/8777026

[17] Chen, W., Wang, Y., & Zhang, X. (2020). A Novel Multi-Model Deep Learning Approach for Collaborative Filtering. Journal of Intelligent & Fuzzy Systems, 39(5), 6565-6574. doi: 10.3233/JIFS-189559

[18] Maloth, B., Suman, J., Saritha, G., & Chandrasekhar, A. (2012). Non linear programming computation outsourcing in the cloud. Int. J. Comput. Sci. Eng. Technol., 2(3).

[19] Chen, W., Wang, Y., & Zhang, X. (2021). Multi-Model Deep Learning Approach to Collaborative Filtering. Journal of Intelligent & Fuzzy Systems, 41(1), 1301-1310. doi: 10.3233/JIFS-201898

[20] Suneel, Chenna Venkata, K. Prasanna, and M. Rudra Kumar. "Frequent data partitioning using parallel mining item sets and MapReduce." International Journal of Scientific Research in Computer Science, Engineering and Information Technology 2.4 (2017).

[21] Rudra Kumar, M., Rashmi Pathak, and Vinit Kumar Gunjan. "Diagnosis and Medicine Prediction for COVID-19 Using Machine Learning Approach." Computational Intelligence in Machine Learning: Select Proceedings of ICCIML 2021. Singapore: Springer Nature Singapore, 2022. 123-133.

[22] N.Satish Kumar & Sujan Babu Vadde (2015). Typicality Based Content-Boosted Collaborative Filtering Recommendation Framework. International Journal Of Computer Engineering In Research Trends, 2(11), 809-813.

[23] Chen, W., Wang, Y., & Zhang, X. (2022). Collaborative Filtering Enhanced by Multi-Model Deep Learning Approach. Journal of Intelligent & Fuzzy Systems, 43(2), 1891-1900. doi: 10.3233/JIFS-219870.

[24] Kumar, P. ., Gupta, M. K. ., Rao, C. R. S. ., Bhavsingh, M. ., & Srilakshmi, M. (2023). A Comparative Analysis of Collaborative Filtering Similarity Measurements for Recommendation Systems. International Journal on Recent and Innovation Trends in Computing and Communication, 11(3s), 184–192. https://doi.org/10.17762/ijritcc.v11i3s.6180.