Bridging Content and Behavior: A Deep Dive into Hybrid Recommendation Systems for Enhancing Personalized User Experiences

by

Amirhossein Ghadami

A thesis submitted to the University of Ottawa
in partial fulfillment of the thesis requirement for the degree of

MASTER OF COMPUTER SCIENCE

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Examining Committee

The following served on the Examining Committee for this thesis.

External Examiner:	Alan Tsang Assistant Professor, School of Computer Science, University of Carleton
Internal Member(s):	Caroline Barrière Assistant Professor, School of Electrical Engineering & Computer Science University of Ottawa
Supervisor(s):	Thomas Tran Professor, School of Electrical Engineering & Computer Science University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those concerning consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

In the rapidly evolving digital landscape, recommendation systems are pivotal in optimizing user experience and driving the economic success of businesses involved in digital commerce and content distribution. These systems intelligently suggest products and services to users by analyzing a myriad of factors such as historical behaviors, item characteristics, and individual preferences. Among the various types, hybrid recommendation systems have emerged as particularly influential. They combine multiple computational techniques to enhance the accuracy of predictions and effectively address the shortcomings inherent in single-method systems by integrating diverse data sources and algorithmic approaches.

This thesis introduces the Convolutional Autoencoder Recommendation System (CAERS), which utilizes a Convolutional Autoencoder (CAE) to deeply analyze and decode content data from users and items. The primary advantage of CAERS lies in its ability to discern intricate content patterns, which significantly aids in mitigating the cold start problem—providing robust recommendations based solely on content analysis, thus enhancing user engagement right from their initial interaction with the system.

Building upon the foundation laid by CAERS, this research extends into the realm of hybrid systems with the development of CAERS-CF. This model synthesizes the strengths of CAERS's deep learning-based content analysis with the traditional collaborative filtering techniques, creating a robust framework that harnesses both the content data and historical user-item interactions. This integration results in a marked improvement in recommendation accuracy, making CAERS-CF a superior choice compared to standalone models and other hybrid configurations.

The culmination of this research is embodied in TriDeepRec, a novel hybrid recommendation system that further extends the capabilities of CAERS by integrating it with Neural Collaborative Filtering (NCF) and a Multilayer Perceptron (MLP). This three-pronged approach allows

TriDeepRec to leverage the nuanced capabilities of CAERS for content data analysis, NCF for insight into behavioral data, and MLP for effectively combining these diverse inputs into a cohesive output. The result is a recommendation system that not only predicts user preferences with high precision but also adapts to new and evolving data patterns.

To quantitatively measure the effectiveness of these innovations, this thesis employs two key metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). These metrics serve to evaluate and compare the performance of our models against established benchmarks. The results clearly demonstrate that our proposed systems offer substantial improvements in recommendation accuracy and reliability, setting new standards for hybrid recommendation systems.

Acknowledgements

Throughout the course of this academic journey, Professor Thomas Tran has stood as a pillar of support and wisdom, guiding me through the complexities of research with patience and profound insight. His calm demeanor and strategic advice provided a steady hand during times of doubt, instilling a sense of confidence and clarity. Professor Tran's approach to mentorship, characterized by deep understanding and thoughtfulness, was instrumental in navigating the numerous challenges that arose during my studies. I am immensely thankful for his invaluable assistance, which transcended the boundaries of academic guidance to include substantial moral and emotional support. His unwavering dedication to nurturing his students' academic and personal growth, coupled with his extensive expertise, significantly influenced the direction and success of my thesis. The assistance provided by Professor Tran was far more significant than I could have initially anticipated, transforming this demanding academic endeavor into a profoundly enriching experience.

On a deeply personal note, I must express my heartfelt gratitude to my parents, Sakineh and Mohammad. Their unconditional love, support, and encouragement have been the foundation of my academic pursuits. It is their steadfast belief in my potential, even during challenging times, that has continuously fueled my determination and ambition. The sacrifices they have made and their relentless support have not only enabled me to chase my dreams but have also been a constant source of motivation and resilience. Their role in my academic and personal achievements is immeasurable, and I owe them an immense debt of gratitude that words can scarcely convey. My parents' influence and sacrifices have shaped me into the person I am today, and for that, I am forever grateful.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of Thesis

As digital environments become increasingly saturated with a vast array of content and products, the need for sophisticated recommendation systems has never been more critical. These systems enhance user experience by personalizing content delivery and product recommendations, thus addressing the challenge of information overload in digital spaces.

Chapter 1: Introduction In this chapter, we outline the motivation and problem statements that drive our research. We discuss the specific contributions of our work to the field of recommendation systems and list the publications and submissions related to our research. This structural overview sets the context for the detailed exploration and innovation presented in subsequent chapters.

Chapter 2: Background, Related Works and Literature Review This chapter delves into the essential background and related works necessary for understanding advanced recommendation systems, focusing on deep learning and hybrid approaches. We begin with a comprehensive literature review that explores the evolution of recommendation systems, from traditional methods to the latest advancements in deep learning technologies.

Chapter 3: CAERS – Convolutional Autoencoder Recommendation System Chapter 3 is dedicated to detailing CAERS, a system that applies deep learning techniques to analyze user-item relationships more intricately. The architecture and functionality of CAERS are explored, highlighting its ability to extract meaningful features from content data, thereby enhancing the accuracy of recommendations.

Chapter 4: CAERS-CF In Chapter 4, we introduce CAERS-CF, a hybrid model that synergizes the deep learning capabilities of CAERS with the established methods of collaborative filtering. This integration allows the model to combine the strengths of both approaches, significantly improving the precision and reliability of the recommendations.

Chapter 5: TriDeepRec Chapter 5 focuses on TriDeepRec, an advanced system that integrates CAERS with Neural Collaborative Filtering (NCF) and a Multilayer Perceptron (MLP). This chapter evaluates how TriDeepRec leverages multiple deep learning models to achieve superior prediction accuracy and enhance the system's overall effectiveness.

Chapter 6: Conclusion and Future Works The final chapter summarizes the research findings and discusses the implications of our work for future developments in recommendation systems. It also outlines potential avenues for further research, aiming to continue the advancement

of recommendation technologies.

Throughout this thesis, the methodologies employed and the performance of the proposed models are evaluated against key metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). This analysis not only demonstrates the effectiveness of our innovative models but also underscores the potential of hybrid recommendation systems to transform user interactions within digital ecosystems.

By exploring and integrating cutting-edge technologies, this thesis contributes to the field of recommendation systems, proposing new models that address the limitations of traditional methods and enhance the accuracy and user experience in digital environments.

1.2 Motivations

Recommendation systems are vital in today's digital landscape, helping users navigate the vast array of content and products available online. These systems tailor recommendations using different types of data: some analyze users' past behavior, like purchase history or browsing patterns, while others examine user or item attributes such as genre or specifications.

This thesis aims to exploit the strengths of both data types by using advanced deep learning techniques to integrate them. Deep learning is celebrated for its versatility across numerous domains, its ability to process substantial datasets, and its efficacy in identifying complex patterns. By applying deep learning, this work seeks to develop a robust recommendation system that combines detailed item attributes with insights into user behavior to offer precise and personalized recommendations. This approach not only aims to improve overall system performance and user satisfaction but also addresses the cold start problem, enhancing the system's ability to make recommendations for new users or items with limited historical data.

Further exploration of these deep learning methods and their integration into a hybrid recommendation system will be detailed in the following chapters. This approach underscores a significant shift towards more sophisticated, adaptive, and user-focused recommendation systems.

The motivation for this thesis stems from the limitations observed in traditional recommendation systems and the potential of deep learning to overcome these challenges. Deep learning models excel at extracting latent preferences and detailed item features, thus enriching the understanding of complex user-item interactions [35]. Convolutional Autoencoder Recommendation System (CAERS) introduced in this thesis leverages these capabilities to enhance recommendation accuracy significantly.

Although promising, deep learning approaches also introduce new challenges, prompting a shift towards hybrid recommendation systems. These systems combine various methods, including deep learning, to provide more robust and accurate recommendations [5]. By drawing on multiple sources of data and methodologies, hybrid systems can deliver superior recommendations that address the inherent limitations of singular approaches [35].

By integrating content data, which provides specific item or user attributes, with past behavioral data, which offers insights into user preferences based on their activity, hybrid systems like CAERS-CF and TriDeepRec achieve a high level of recommendation accuracy. This synthesis allows these systems to leverage the best aspects of both content-based and behavioral data, resulting in highly effective and tailored recommendations.

In conclusion, this thesis contributes to advancing recommendation systems by leveraging deep learning and hybridization techniques to merge insights from both content and behavioral data [35].

1.3 Problem Statement

How can content information be combined with historical data?

In the realm of recommendation systems, a significant challenge is to effectively harness both content data and past behavioral data to improve the quality of recommendations. Traditional systems typically focus on one of two approaches: content-based methods, which analyze item attributes to suggest similar items, or collaborative filtering methods, which predict user preferences based on similar user behavior patterns. Each approach has its merits, but often they do not fully capture the complexities and subtleties of user preferences when used in isolation.

The goal of integrating these different types of data is to overcome the limitations inherent in each individual method. By combining content-based and collaborative filtering techniques, it becomes possible to provide a more comprehensive and accurate prediction of what users might prefer. This integration not only enhances the relevance of the recommendations made but also enriches the user's experience by offering suggestions that are finely tuned to their unique tastes and previous interactions.

However, the process of integrating these approaches presents its own set of challenges. It involves the complex task of hybridizing models in a way that effectively leverages the strengths of both content and behavioral data. This requires sophisticated algorithms capable of parsing through vast amounts of diverse data and extracting meaningful patterns. The hybrid models must be designed to seamlessly blend the insights gained from both data types, ensuring that the complementary strengths of content-based and collaborative approaches are maximized. This strategic fusion aims to create a more robust and adaptable recommendation system, capable of delivering superior performance in predicting user preferences, thereby enhancing overall user satisfaction and engagement.

The primary challenge in this context is identifying the optimal strategy to merge content-

based and behavior-based recommendation models to minimize prediction errors. This involves not only the technical complexity of processing and analyzing heterogeneous data sources but also the algorithmic innovation required to seamlessly combine the insights gleaned from each model type. The goal is to develop a hybrid system that can accurately predict user preferences with lower error rates, thereby improving the relevance and personalization of recommendations. Achieving this requires overcoming obstacles related to model compatibility, data scalability, and the dynamic nature of user-item interactions, making the quest for an effective hybrid recommendation system a significant pursuit in the field.

1.4 Research Contributions

In this thesis, we introduce three significant contributions to the field of recommendation systems, each addressing different facets of the challenge of effectively combining content data with user behavior patterns to improve recommendation accuracy and relevance. These contributions are as follows:

- **CAERS: Convolutional Autoencoder Recommendation System:** CAERS introduces a novel paradigm within content-based recommendation systems by employing a convolutional autoencoder. This approach leverages the sophisticated feature extraction capabilities of convolutional neural networks to analyze and process both items' and users' content data effectively. By extracting and utilizing the nuanced features from detailed descriptions or attributes of items and profiles of users, CAERS is adept at predicting user preferences accurately. This capability is particularly beneficial in scenarios where there is no prior user-item interaction data, commonly referred to as the cold start problem. CAERS is specifically engineered to address these challenges, ensuring that new users or items are

seamlessly integrated into the recommendation process, thereby enhancing the user experience right from the start.

- **CAERS-CF: CAERS with Collaborative Filtering:** Building on the capabilities of CAERS, the CAERS-CF model blends content-based filtering with traditional collaborative filtering techniques. This integration combines the deep learning analysis of content data from CAERS with a collaborative filtering approach that utilizes past user behavior through linear regression. As a result, CAERS-CF offers a more detailed understanding of user preferences. This hybrid model improves the accuracy of recommendations by considering both the specific qualities of items and the patterns of user interactions. CAERS-CF effectively merges content relevance with behavioral insights, providing recommendations that are both accurate and personalized. By harmonizing these different approaches, CAERS-CF enhances the overall effectiveness of the recommendation process, making it more adaptable to various user needs and preferences.

- **TriDeepRec: Hibridized CAERS with two other deep learning methds**

  TriDeepRec represents the culmination of our research efforts, embodying a sophisticated hybrid recommendation system that synergizes the strengths of CAERS, Neural Collaborative Filtering (NCF), and an additional layer of deep learning integration through a Multilayer Perceptron (MLP). This advanced model not only capitalizes on the benefits of analyzing both item and user content as well as user behavior, but it also employs deep learning to intricately fuse these insights, resulting in lowered prediction errors. TriDeepRec sets a new benchmark in the recommendation system domain by offering a more accurate recommendation system compared to state-of-the-art models.

Each of these contributions showcases the potential of deep learning and hybrid models in

revolutionizing recommendation systems, offering insights into the effective integration of diverse data types to enhance user experience.

1.5 Publications

The work of this thesis has resulted in three journal submissions, as follows:

1. **Ghadami, A.**, & Tran, T. (2024). *Convolutional Autoencoder Recommendation System*. Submitted to International Journal of Data Science and Analytics.

2. **Ghadami, A.**, Tran, T. (2024). *CAERS-CF: Enhancing Convolutional Autoencoder Recommendations through Collaborative Filtering*. Submitted to Knowledge and Information Systems. First submission: February 2024. Revision submission: June 2024.

3. **Ghadami, A.**, & Tran, T. (2024). *TriDeepRec: A Hybrid Deep Learning Approach to Content and Behaviour-based Recommendation Systems*. Submitted to User Modeling and User-Adapted Interaction. First submission: March 2024. Revision submission: June 2024.

Chapter 2

Background Information, Related Work and Literature Review

2.1 Introduction

This chapter offers a detailed overview of the foundational concepts, previous research, and a thorough review of existing literature in the field of recommendation systems. By examining and synthesizing the key findings from previous studies, this chapter sets the stage for understanding the complexities and innovations discussed in this thesis

2.2 Background Information

This section provides an exploration of the various dimensions of recommendation systems, beginning with an overview of the concept and its pivotal role across multiple industries. We

start by delving into the fundamentals of recommendation systems, explaining how they function and their importance in enhancing user experiences through personalized content suggestions.

Following this, we discuss deep learning and its extensive applications across various domains, highlighting its ability to handle complex and voluminous data sets effectively. This segment underlines how deep learning techniques have revolutionized fields such as image and speech recognition, and how these advancements contribute to the development of more sophisticated recommendation systems.

Finally, we examine convolutional autoencoders, a specialized form of deep learning architecture, and explore their contributions to the field of recommendation systems. Convolutional autoencoders are particularly noteworthy for their efficiency in feature extraction and dimensionality reduction, making them well-suited for dealing with image-based data. This structured approach will elucidate the evolution of recommendation systems, demonstrating how they have adapted to address increasingly complex data and user demands, thus providing a thorough understanding of the current landscape and future potential of these technologies.

2.2.1 Recommendation Systems

A recommendation system is a type of information filtering system that uses algorithms to suggest items to users based on various criteria, including user preferences, behavior, and item characteristics. These systems are essential for many businesses as they help personalize user experiences, increase user engagement, and optimize product or service inventories. For example, e-commerce platforms like Amazon enhance shopping experiences by suggesting products that customers might like based on their previous searches, purchases, and browsing behavior. This not only helps customers discover products they might not have found on their own but also significantly increases sales through cross-selling or upselling related items. Streaming services

such as Netflix [2] and Spotify [4] use recommendation engines to suggest movies, shows, and music based on viewing or listening histories, ratings, and the preferences of similar users. This personalization improves user satisfaction and retention, encouraging longer engagement with the platform. Social media platforms like Facebook and Twitter recommend content, pages, or people to follow based on a user's interactions, such as likes and shares, and the behavior of similar users. This helps enhance engagement by ensuring users are exposed to content that interests them. Financial services, including investment platforms and banks, recommend products like mutual funds or credit cards based on user spending habits, investment history, and risk profiles [34]. This tailored approach helps institutions better meet their clients' needs and strengthen client relationships.

Online advertising also extensively utilizes recommendation systems to target ads more effectively. By analyzing user behavior, preferences, and demographic data, businesses can serve personalized ads that are more likely to engage users, enhancing the effectiveness of advertising campaigns.

2.2.2 Deep Learning

Deep learning, a subset of artificial intelligence, employs layered neural networks to analyze and interpret large amounts of data. These networks consist of multiple layers of nodes, resembling the neural connections in the human brain, that process and learn from data inputs to recognize patterns and make informed predictions. This technology excels in handling tasks that involve unstructured data such as images, text, and audio, allowing it to effectively solve problems that are too complex for traditional algorithms.

In the field of recommendation systems, deep learning enhances the ability to provide personalized content and product suggestions. For instance, it powers sophisticated recommendation

engines that analyze user preferences, past interactions, and contextual information to suggest relevant items in e-commerce platforms or streaming services. Additionally, in conversational systems, deep learning is instrumental in developing chatbots and virtual assistants that can understand and respond to user queries with high relevance, creating dynamic, engaging, and contextually appropriate interactions. The versatility of deep learning in processing diverse data types and its application in both static and interactive environments demonstrate its broad utility and transformative potential across various sectors [8].

2.2.3 Convolutional Autoencoder

A convolutional autoencoder [46] is a specialized type of neural network that merges the features of convolutional neural networks (CNNs) [23] and autoencoders [3], making it highly effective for tasks involving image data such as feature learning and dimensionality reduction. Initially developed for image processing tasks like denoising and compression, convolutional autoencoders excel in handling spatial hierarchies where patterns are more structured.

The operation of a convolutional autoencoder involves two primary phases: encoding and decoding. In the encoding phase, the input image passes through several convolutional layers, each applying a set of learned filters or kernels [20] that reduce the dimensionality while capturing essential features. This is achieved without fully connected layers, allowing the model to maintain spatial hierarchies and local features. The decoding phase then aims to reconstruct the original input from this encoded representation using upsampling techniques and convolutional layers to gradually increase the resolution of the output to match the original input size, thereby focusing on the essential features captured in the encoded data.

The convolutional layers, which perform the convolution operations, are key components. Convolutional operation is a mathematical process where a filter (or kernel) is applied to the

input matrix. The filter slides across the input data, performing element-wise multiplication and summing the results to produce a single value in the output matrix. This process is repeated across the entire input, generating feature maps that recognize patterns such as edges and textures. Kernels or filters [20] convolve with the input matrix, producing these feature maps.

Pooling layers in the encoder reduce the dimensionality of each feature map while retaining important information through methods like max pooling. In the decoder, upsampling layers increase the spatial dimensions to assist in reconstructing the image. Activation functions like Rectified Linear Unit (ReLU) or sigmoid introduce non-linear properties to the network, enabling it to learn complex patterns.

Kernels in these layers are small matrices that extract features from the images, automatically learned during training. The kernel size influences the granularity of features captured, with smaller kernels picking up fine details and larger ones focusing on more global features [20].

The versatility of convolutional autoencoders is thus not only confined to image compression and reconstruction but also extends to advanced applications such as anomaly detection, sophisticated image manipulation, text processing, and enhancing recommendation systems. This broad applicability underscores the importance of CAEs in modern data processing tasks across various domains.

2.3 Literature Review

Recommendation systems have long served as essential tools in digital environments, guiding users through vast information to identify content that aligns with their preferences. Initially grounded in traditional methods, these systems encompass three main types: those that analyze user behavior to predict preferences based on historical interactions, those that focus on item

attributes, and hybrid systems that combine both approaches. Each type leverages distinct data sources and methodologies to provide personalized suggestions [25, 32].

Behavior-based systems, particularly those using collaborative filtering, scrutinize user interactions such as ratings and reviews to identify patterns among similar users, enabling the recommendation of mutually liked items [5,36]. Collaborative filtering, a key method in behavior-based systems, predicts user preferences either through memory-based approaches, which directly use historical interactions to find user or item similarities, or model-based approaches, which involve training a model on historical data to learn these patterns [18]. These methods can effectively address user preferences but often struggle with the cold start problem when new users join and face data sparsity issues with limited user interactions.

Conversely, content-based systems recommend items by matching the characteristics of previously liked items, using details such as genre or author [35]. This approach is beneficial when extensive metadata is available, allowing the system to function effectively even with minimal user interaction data. However, its reliance on a user's established preferences may restrict the diversity of recommendations, potentially leading to a narrower user experience.

Hybrid systems integrate the strengths of both behavioral and attribute-based methods to enhance recommendation accuracy and versatility [1, 15]. By combining user behavioral patterns with item attributes, they offer more dynamic and comprehensive recommendations. This versatility is especially beneficial in managing the cold start problem and adapting to varied user and item contexts.

Understanding these foundational recommendation systems sets the stage for advancements that incorporate deep learning, which has revolutionized many fields by processing large-scale data and uncovering intricate patterns [22, 44]. Deep learning's multi-layered algorithms, which mimic the human brain's learning process, allow for the abstraction of complex data hierarchies.

These capabilities enable deep learning-based systems to handle the high dimensionality of user-item data and the scalability issues traditional systems face [21].

Deep learning-based recommendation systems employ sophisticated architectures that can process vast arrays of information efficiently, transforming sparse data into dense vectors and utilizing contextual information to enhance the relevancy and personalization of recommendations [6]. Furthermore, hybrid recommendation systems that incorporate deep learning not only tackle traditional system limitations but also improve recommendation diversity and user satisfaction by seamlessly integrating multiple predictive models [15].

Our hybrid approaches, which blends content-based features with user interaction data and deep learning techniques, offers a robust solution to the challenges faced by traditional recommendation systems. By effectively harnessing the strengths of each component, these advanced systems deliver more accurate and relevant recommendations, significantly enhancing user engagement and satisfaction.

2.4 Related Works

2.4.1 Introduction

This section of the thesis presents a detailed review of the literature and studies related to recommendation systems, with a specific focus on the advancements and methodologies that have shaped this field. By examining the works of other researchers and the evolution of recommendation technologies, we aim to contextualize our research within the broader academic and practical frameworks.

2.4.2 Deep Learning-Based Recommendation Systems

Deep learning models, with their sophisticated architectures, have demonstrated remarkable proficiency in extracting essential features from data, significantly improving recommendation accuracy and effectiveness.

The GAP model [24] employs a convolutional neural network (CNN) to process an outer product matrix of features, which is formed by combining user and item embeddings. The use of cross-convolutional filters allows the GAP model to capture intricate user-item interactions and emphasize critical features while minimizing overfitting through global pooling. Despite its strengths, the GAP model's reliance on an outer product matrix can introduce computational overhead, especially in scenarios with large-scale datasets.

In contrast, the GCF-YA model [43] addresses the challenge of data sparsity by incorporating a deep graph neural network (GNN) with information propagation and an attention mechanism for link prediction. This model excels at capturing complex relationships in sparse data by utilizing a graph-based approach where each node represents either a user or an item. However, the GCF-YA model's complexity and graph-based nature may pose scalability issues with very large datasets.

Deep Matrix Factorization (DMF) [41] utilizes a ratings matrix to input data into two separate multi-layer perceptrons (MLPs) that process user and item information independently. The predicted ratings are derived by computing the cosine similarity between the feature vectors produced by these MLPs. While DMF effectively captures latent user and item features, its relatively shallow architecture may limit its ability to model complex interactions compared to more sophisticated deep learning models.

The DNNRec model [17] features an embedding layer followed by a multi-layer perceptron. It begins by embedding categorical data into vectors, which are then processed through the MLP

to predict user ratings. DNNRec integrates side information with user and item embeddings to enhance predictive accuracy. However, it may struggle with highly complex user-item interactions, unlike models that use convolutional layers to capture intricate patterns.

The Deep & Cross Network Modified (DCN-M) [38] improves upon the original Deep & Cross Network by more effectively learning feature interactions, which is particularly advantageous in large-scale commercial settings. This model enhances scalability and performance, but it may not fully address the sequential dependencies and contextual information that are crucial in some recommendation tasks.

Recent advancements have seen the use of convolutional neural networks to analyze user-item dynamics. For instance, Convolutional Neural Collaborative Filtering with Outer Products (ConvNCF) introduces a technique that combines user and item identifiers to create an interaction matrix through an outer product. This matrix is processed by a convolutional layer designed to capture detailed patterns of user-item interactions [13].

Convolutional Factorization Machines (CFM) integrate multiple crucial layers: an embedding layer, a self-attention pooling layer, and a convolutional layer that processes content information. The system emphasizes feature significance through an attention mechanism, followed by pooling, and analyzes interactions via an outer product of feature pairs. These interactions are then processed using a 3D convolutional filter to detect complex signals [40].

CAERS (Convolutional Autoencoder-based Recommendation System) employs a convolutional autoencoder to process item content and combines it with collaborative filtering to enhance recommendation accuracy. Unlike the GAP model or ConvNCF, which primarily focus on user-item interactions without incorporating content-based information, CAERS integrates both content-based and collaborative filtering methods. This integration helps CAERS manage data sparsity more effectively and provides a richer representation of items compared to DMF and

GCF-YA, which may not handle sparse data as robustly.

2.4.3 Hybrid Recommendation Systems

These systems represent advanced hybrid recommendation approaches, leveraging multiple techniques to enhance accuracy and handle data sparsity.

The Enhanced Collaborative Autoencoder (ECAE) [28], an advancement of the collaborative denoising autoencoder (CDAE) [39], uses ratings as its main input. It addresses the issue of sparse data by incorporating a generative network designed to handle complex binary targets and produce soft targets. Predictions are made through a weighted combination of outputs from the generative network and a retraining network. The latter, operating as an autoencoder, uses these soft targets to refine its performance. ECAE's complexity may increase computational costs, but it effectively addresses sparse data issues.

Neural Collaborative Filtering (NCF) [14] merges traditional matrix factorization [19] with modern neural network approaches. It utilizes feature vectors derived from matrix factorization as inputs to a multi-layer perceptron, enabling the modeling of complex, high-dimensional user-item relationships. Although NCF improves upon traditional matrix factorization, it does not fully incorporate content-based or side information, limiting its scope compared to more integrated hybrid approaches.

The Correlative Denoising Autoencoder (CoDAE) [29] is designed specifically for social networks, employing separate autoencoders for handling ratings, truster, and trustee matrices. This approach helps manage data sparsity and enhances feature correlation learning through a shared weight matrix and a specialized regularization term that captures interactions across the three different data types. While CoDAE effectively manages data sparsity and feature correlations, its specialization in social networks may limit its applicability to other contexts.

TriDeepRec combines content-based and behavior-based data through deep learning techniques, specifically merging CAERS and NCF architectures via an MLP. This hybrid approach enhances recommendation accuracy by integrating collaborative and content-based filtering methods, overcoming the limitations of ECAE and NCF, which primarily focus on one type of data. TriDeepRec's ability to learn from both user behaviors and content features provides a more comprehensive solution compared to CoDAE, which is specialized for social networks.

2.4.4 Convolutional Autoencoder-Based Recommendation Systems

Several studies have explored the potential of Convolutional Autoencoders (CAEs) in recommendation systems. The Collaborative Convolution AutoEncoder (CCAE) [45] tackles the common issues of sparsity and accuracy in traditional collaborative filtering. This method uses convolutional autoencoder techniques to develop recommendation scores, starting with input data drop sampling followed by compression through convolution and downsampling, and concluding with reconstruction via deconvolution and upsampling.

Similarly, the Collaborative Convolutional Autoencoder (CCA) [37] aims to enhance recommendation systems, particularly in Collaborative Filtering. The CCA method enriches Matrix Factorization by integrating content features into user-item matrices. This approach leverages Sentence-BERT (SBERT) [31] and a Convolutional Neural Network (CNN) [23] to process contextual information from textual data.

CAERS integrates a convolutional autoencoder with collaborative filtering to process item content and improve recommendation accuracy. Unlike CCAE and CCA, which focus on either content-based or collaborative filtering techniques individually, CAERS combines both approaches, making it more effective in managing data sparsity and providing accurate recommendations. CAERS also addresses the computational challenges associated with convolutional

methods by efficiently integrating content and collaborative data, offering a more balanced and comprehensive approach compared to the standalone models.

2.4.5 Collaborative-based Recommendation System

Collaborative filtering operates on the principle of analyzing interaction data between users and items to predict user preferences. This traditional recommendation technique utilizes historical user behavior data to make predictions [20, 32].

There are two main types of collaborative filtering: Memory-Based and Model-Based approaches. Memory-Based collaborative filtering is subdivided into User-Based and Item-Based methods [18]. In User-Based collaborative filtering, the system estimates how a user might rate a new item by analyzing the ratings of similar items by users within the same neighborhood. If these similar users generally rate the new item positively, it is likely to be recommended [33]. Conversely, Item-Based collaborative filtering predicts ratings based on the similarity between items rather than users. It calculates a weighted average of ratings that a user has given to similar items previously [33].

Model-Based collaborative filtering, unlike its Memory-Based counterpart, does not directly utilize raw historical data. Instead, it involves building and training a model on this data to identify underlying patterns and relationships [18].

Our research builds upon these established collaborative filtering frameworks by incorporating advanced deep learning techniques. This integration enhances traditional collaborative filtering processes, improving both accuracy and functionality in our recommendation system.

For instance, the ExtKNNCF model represents an advanced form of KNN-based collaborative filtering [26]. This model adapts the traditional KNN approach by considering user cognitive

processes, thereby enhancing the personalization and accuracy of recommendations. It dynamically adjusts user clusters based on cognitive similarity, effectively addressing challenges such as the cold start problem that impacts new users.

Another notable method, URP-CF-SIM, explores the effectiveness of user-user and item-item collaborative filtering in e-commerce contexts [27]. This method compares different collaborative filtering techniques, including user K-Nearest Neighbor (K-NN) using cosine similarity and Pearson correlation, and item-based K-NN with traditional and matrix-based approaches like Matrix Factorization (MF). The results demonstrate that item-item K-NN with Pearson correlation performs particularly well, underscoring its capability to provide highly personalized recommendations.

offers a significant advantage over traditional and advanced collaborative filtering models by effectively integrating content-based information with collaborative filtering techniques. Unlike memory-based methods, which struggle with data sparsity and cold start problems, or model-based approaches that rely heavily on user-item interaction data, CAERS-CF utilizes a convolutional autoencoder to process item content, thereby enhancing its ability to make accurate recommendations even with sparse data. This integration allows CAERS-CF to capture complex patterns and interactions between users and items more comprehensively, providing a richer and more robust recommendation system compared to models like ExtKNNCF and URP-CF-SIM, which either focus on cognitive processes or traditional collaborative filtering techniques without incorporating content-based features.

Chapter 3

Convolutional Autoencoder Recommendation System (CAERS)

3.1 Introduction

In this chapter, we delve into the implementation of our deep learning recommendation system, known as the Convolutional Autoencoder Recommendation System (CAERS). CAERS is based on the Convolutional Autoencoder (CAE), which effectively merges the structural advantages of Convolutional Neural Networks (CNNs) [23] with the data-processing capabilities of autoencoders [3].

Convolutinal Autoencoder is particularly effective at handling grid-like data structures, such as images, and features two main components: an encoder that compresses the data to a more manageable form, and a decoder that reconstructs the original data from this compressed format [3]. One of the key strengths of CAEs is their flexibility; they do not require that the input data exactly match the output. This characteristic is invaluable in scenarios like anomaly detection,

where the system is specifically tuned to accurately reconstruct normal instances but will struggle to do so with anomalous data, effectively identifying outliers [9].

Our research leverages the Convolutional Autoencoder (CAE) architecture to refine and enhance recommendation systems. By utilizing this advanced framework, we aim to uncover and understand the intricate dynamics that exist between users and items. This involves gathering detailed data about both, such as item descriptions, and employing sophisticated embedding techniques specifically designed for our model's requirements. These processes prepare the data for integration into our dataset. Subsequently, we create a user-item interaction matrix by calculating the dot product of embedded user and item data, which provides a structured way to analyze interactions.

In developing our CAE model, we innovate beyond traditional autoencoders by using this user-item interaction matrix as our input and aiming to output a corresponding ratings matrix. Through this approach, the encoder captures the nonlinear, high-order interactions among users and items. Then, the decoder translates these interactions into a format of ratings. This novel method enables us to effectively identify and model the complex relationships embedded within the data, thus enhancing the accuracy and efficacy of our recommendation system.

A critical aspect of our approach is addressing the cold start problem, which occurs when new items or users are introduced into the system. Unlike conventional methods that heavily depend on historical interaction data, our strategy emphasizes content information. This focus allows our system to effectively manage the cold start problem by providing robust recommendations for new users and items from the outset.

This chapter contributes significantly to the field in two main ways:

- It introduces an innovative application of Convolutional Autoencoders to detect and model the complex nonlinear relationships within user-item data, which enhances the accuracy

and relevance of the recommendations provided.

- It offers a practical solution to the cold start problem by prioritizing content information, ensuring that new users and items are seamlessly integrated into the recommendation system without the need for extensive historical data.

3.2 CAERS Methodology

This section outlines our methodical approach, utilizing a Convolutional Autoencoder (CAE) to analyze the nuanced interactions between users and items. The process involves three fundamental steps, each aimed at enhancing our system's ability to understand and predict user preferences with greater accuracy:

1. **Data Collection:** Our first step involves the systematic gathering of essential information about users and items. This data is crucial as it forms the foundation of our analysis. We integrate this new information into our existing dataset, ensuring we have a comprehensive base from which to work.

2. **Feature Embedding:** After collecting the data, we then move to embed the various attributes of users and items into the dataset. This step involves transforming these attributes into a format that can be effectively processed by our model. Embedding allows us to condense the rich information into a manageable, structured form that highlights significant patterns and characteristics.

3. **Model Development and Training:** The final step is to construct and train the CAE model. This model is equipped with convolutional layers that utilize different sizes of kernels to identify and learn from the complex patterns found in the user-item interactions.

These layers are critical as they allow the model to capture the non-linear relationships that traditional analytical methods might overlook.

By following these steps, we enhance our model's capacity to discern the intricate dynamics of user-item interactions. This approach not only improves the accuracy of our predictions but also deepens our understanding of how different elements within the data interact. As a result, our recommendation system becomes more adept at suggesting items that align more closely with individual user preferences, thereby increasing the personalization and effectiveness of the recommendations provided.

3.2.1 Data Extracting and preprocessing

The effectiveness of deep learning models in recommendation systems is highly contingent on the quality and depth of the data utilized, especially when it involves analyzing interactions between users and items. In our study, we focus on using movie datasets, which allow us to evaluate the model's performance in a controlled and information-rich environment.

To ensure a robust dataset, we collect detailed information about each movie from a variety of sources, with a significant portion coming from Wikipedia. For data collection, we utilize Python libraries such as `BeautifulSoup` and `Requests`. The `Requests` library is instrumental in sending HTTP requests to fetch the HTML content of Wikipedia pages. Once we have the HTML content, `BeautifulSoup` is employed to parse it and extract relevant information, such as movie plots, character descriptions, and thematic elements. This process involves identifying and extracting the sections of the page that contain the desired textual data, allowing us to compile comprehensive movie descriptions that include main characters, pivotal challenges, underlying motives, and plot resolutions.

After collecting the data, it is crucial to preprocess the textual content to prepare it for training the deep learning models. We employ the `Natural Language Toolkit (NLTK)` library for this purpose. Here's how NLTK is applied in our context:

Tokenization: This process involves breaking down the extracted text from Wikipedia into individual words or tokens. For instance, the plot description of a movie is split into tokens like "The", "movie", "is", "about", "a", "scientist", etc. Tokenization is crucial because it transforms continuous text into discrete units that can be analyzed more effectively by the model.

Stop-word Removal: After tokenization, we remove common words that do not carry significant meaning or contribute to the model's understanding. Stop-words such as "the", "is", and "and" are filtered out. This step is important for reducing noise in the data and focusing the model on the more informative parts of the text.

Lemmatization and Stemming: These techniques are used to reduce words to their base or root forms. Lemmatization involves mapping words to their dictionary forms (e.g., "running" becomes "run"), while stemming reduces words to their root forms (e.g., "running" becomes "run" and "runner" becomes "run"). Both techniques help in consolidating different forms of a word into a single representative form, thereby reducing redundancy and improving the model's ability to recognize and learn from the text.

By integrating these preprocessing steps, we ensure that the text data is normalized and structured in a way that enhances the model's learning capabilities. The cleaned and tokenized data, stripped of unnecessary variations and irrelevant words, provides a more focused and consistent input for the deep learning algorithms.

This combined process of data collection and preprocessing significantly enhances the dataset's quality by expanding its scope and adding layers of contextual depth. This enriched dataset serves as a solid foundation for the subsequent training phase, ensuring that our model can learn

from a detailed and comprehensive dataset.

An example of the type of descriptive data we compile is shown below in Figure 3.1.

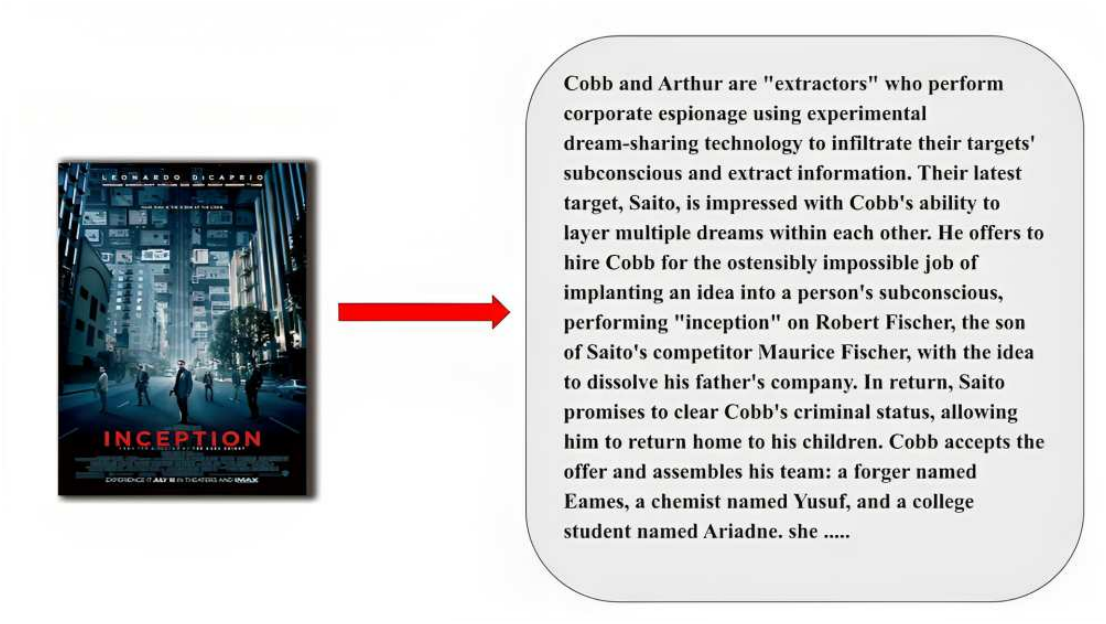

Figure 3.1: Example of Movie (Inception 2014) Description

3.2.2 Embedding

In our dataset, we deal with two principal types of variables: continuous and categorical. Continuous variables, such as height or temperature, can take any value within a range, continuously spanning from one value to another. On the other hand, categorical variables categorize data into discrete groups, like a user's profession, and are typically represented in a model using distinct categories.

Handling categorical data often involves converting these categories into dummy variables, a process that results in a sparse matrix where each category is represented by a binary vector.

This method, while straightforward, can obscure the underlying relationships between categories due to the increased sparsity.

To address this, we employ an embedding technique that compresses the sparse, high-dimensional data of dummy variables into a dense, lower-dimensional, and more informative feature space. This transformation is facilitated by an embedding table, a learned weight matrix that maps each category to a continuous vector that captures more nuanced information about the category. The embedding process for a categorical variable is described mathematically in Equation (3.1):

$$v_i^T = o_i^T V \tag{3.1}$$

Here, v_i^T represents the embedding vector for the i-th category, V denotes the embedding matrix, and o_i is the one-hot encoded vector of the i-th category. The parameters m and n in the matrix $V \in \mathbb{R}^{m \times n}$ stand for the number of categories and the dimensionality of each embedding vector, respectively.

To make our embeddings more meaningful and useful, especially for continuous variables, we use a Multi-Layer Perceptron (MLP). This approach helps us keep and even enhance the context of each variable, as the MLP looks at not just individual features but also how they interact with each other. This method is better at capturing complex relationships and reducing the loss of information that simpler encoding methods might miss.

Here's how it works: the variables are first turned into an embedding through this process. These embeddings are then combined into a single, unified representation. This combined embedding effectively captures a detailed profile of both users and items, which is crucial for the next steps in our model. By applying this process iteratively across all variables, we ensure that every part of our dataset is represented uniformly and meaningfully.

So, while the embedding MLP process is a key part of our model, it's not an isolated step.

It's integrated into the overall system, where the refined embeddings feed into subsequent layers or mechanisms to generate final recommendations or insights. This approach not only improves the quality of our embeddings but also makes sure they are well-suited for the rest of the model's analysis.

3.2.3 Building and Training the Convolutional Autoencoder

The Convolutional Autoencoder (CAE) model we have developed combines the strengths of Convolutional Neural Networks (CNN) [23] with the foundational principles of traditional Autoencoders [3]. This integration aims to harness the detailed feature extraction capabilities of CNNs while leveraging the data compression and reconstruction abilities of Autoencoders. This synergy is particularly effective in enhancing the model's proficiency in identifying crucial features within the data, which is vital for overcoming typical challenges such as overfitting, often encountered in complex datasets.

User-Item Matrix Formation Following the extraction and embedding of user and item content data, we proceed to construct a user-item interaction matrix, which is crucial for feeding into the CAE model. This matrix is formed by calculating the dot product between the embedded representations of users and items. Specifically, we use U to represent the matrix of embedded user content, and I for the matrix of embedded item content (examples are provided in Tables 3.2 and 3.3, respectively). The resulting user-item matrix, denoted as D, is derived as follows:

$$D = U \cdot I^{\top} \tag{3.2}$$

This equation signifies the interaction matrix D as the product of user embeddings U and the transpose of item embeddings I, capturing the potential affinities between users and items based

on their embedded features.

Now this matrix serves as the input for the CAE, facilitating the subsequent learning processes.

	Age	Occupation	Sex
U1	28	Sales Person	Female
U2	18	Student	Male
U3	.....	.....	.....
U4	.....	.....	.....

Figure 3.2: User Matrix

	Movie Name	Genres	Overwies
I1	Toy Story (1995)	Family/Adventure	Woody (Tom Hanks), a good-hearted cowboy doll who belongs to a young boy named Andy (John Morris), sees his .....
I2	Inception (2010)	Action/Sci-fi	Dom Cobb (Leonardo DiCaprio) is a thief with the rare ability to enter people's dreams and steal their secrets from their subconscious....
I3	.....	.....	.....
I4	.....	.....	.....

Figure 3.3: Item Matrix

Structure of the Convolutional Autoencoder The Convolutional Autoencoder (CAE), as depicted in Figure 3.4, consists of two fundamental components: the encoder and the decoder. The encoder serves as the first half of the autoencoder architecture, where its primary function is to compress and encode the incoming data into a more manageable, condensed form. This is achieved through a series of convolutional layers that are adept at capturing the meaningful information. These layers apply various filters to the input, extracting significant features while reducing data dimensions without losing essential information.

$$\text{Encoded Data} = \text{Encoder}(\text{Input Data}) \tag{3.3}$$

Once the encoding process completes, the decoder takes over with a crucial role that is not data reconstruction. Unlike typical Convolutional Autoencoders, our decoder is specifically tasked with transforming the encoded data into ratings matrix. This transition involves using a series of deconvolutional and upsampling layers to upscale the compressed feature maps, reshaping them into an output structure that aligns with predicted user-item ratings.

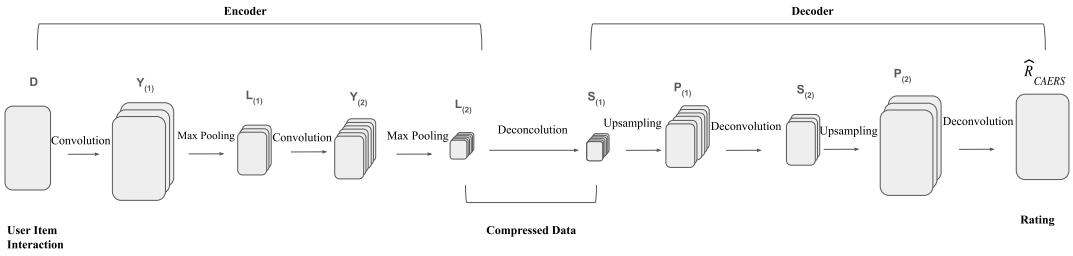

Figure 3.4: Diagram of the CAERS Method

Our design incorporates convolutional layers in both the encoder and decoder parts of our

model. In the encoder, these layers use filters or kernels to extract important features from the input data [23]. These kernels, which are essentially learned weights, move across the input matrix to capture intricate patterns [20]. In the decoder, the layers, known as deconvolutional layers, function differently. While the encoder layers focus on detecting complex patterns and relationships, the decoder layers work on transforming the compressed, encoded outputs back into a ratings matrix. Throughout the training process, our primary aim is to fine-tune these kernels to enhance their ability to process and extract features from large datasets efficiently..

Once the user-item interaction matrix is prepared, it is fed into the first convolutional layer to start the encoding process. The encoding operation applied by the convolutional layers can be mathematically represented as follows:

$$Y_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} D_{i+m-1,j+n-1} \cdot K_{m,n} \tag{3.4}$$

In this formulation, $Y_{i,j}$ denotes the output value located at the (i, j) position in the resulting matrix Y. The operation includes a double summation that cycles through the dimensions M and N of the kernel. Each kernel element $k_{m,n}$ multiplies a corresponding data point from the input $D_{i+m-1,j+n-1}$, with all results cumulatively added together.

It is important to highlight that the input for each convolutional layer can differ depending on its position within the network. Initially, the input might simply be the raw input matrix. Yet, for layers deeper in the network, like the second convolutional layer, the input becomes the output from either the previous convolutional layer or a preceding pooling layer. This flexibility in the input source is critical for accommodating the layered, hierarchical architecture of convolutional neural networks. Each layer refines the output from its predecessors, thereby progressively building up the final output matrix Y. For simplicity in this discussion, we treat D, the user-item interaction matrix, as the initial input in Equation (3.4).

Pooling layers play a vital role in the architecture of Convolutional Neural Networks (CNNs) and Convolutional Autoencoders (CAEs), serving primarily to reduce the spatial dimensions of the input feature maps. This reduction in size, known as downsampling, not only diminishes computational load but also enhances the network's ability to discern critical features within the input data. Pooling achieves this by extracting either the maximum value (max pooling) or the average value (average pooling) from predefined regions of the feature map. Such a mechanism makes the model less sensitive to the exact positions of features within the input, thereby fostering more robust feature detection across varying inputs.

In our methodology, we specifically implement max pooling, which selects the maximum value within each window of the feature map, as described by the following equation:

$$L_{i,j} = \max(Y_{2i,2j}, Y_{2i,2j+1}, Y_{2i+1,2j}, Y_{2i+1,2j+1}) \tag{3.5}$$

Here, $L_{i,j}$ denotes the output of the max pooling operation at position (i, j), and Y symbolizes the output from the convolutional layers immediately preceding the pooling layer. This selective downsampling method effectively highlights the most significant features, simplifying the feature maps while maintaining the most prominent aspects necessary for further processing by subsequent layers of the network.

The decoder plays a vital role in reconstructing the input data shape from the compressed representation obtained by the encoder. This reconstruction process often involves upsampling and transposed convolution operations. Transposed convolution layers, also known as deconvolution layers, are pivotal in the upsampling process. The transposed convolution formula, shown in Equation (3.6), might seem similar to that of a convolutional layer, but it differs significantly in terms of input:

$$S_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} L_{i+m-1,j+n-1} \cdot F_{m,n} \tag{3.6}$$

Where L represents the compressed and encoded data obtained from the encoder phase. The transposed convolution layers use this compressed data as input, differing from typical convolution layers which use either the original input matrix or outputs from prior layers. This allows for the expansion of the data dimensions, effectively 'upscaling' the compressed information.

Subsequent to transposed convolution, the upsampling layers further increase the spatial dimensions of the data. This is performed using a common upsampling technique expressed in Equation (3.7):

$$P_{i,j} = \frac{1}{4} \left(S_{\lfloor i/2 \rfloor, \lfloor j/2 \rfloor} + S_{\lfloor i/2 \rfloor, \lceil j/2 \rceil} + S_{\lceil i/2 \rceil, \lfloor j/2 \rfloor} + S_{\lceil i/2 \rceil, \lceil j/2 \rceil} \right) \tag{3.7}$$

In this final stage, the output of the transposed convolutional layer S serves as the input for the upsampling process, resulting in P, which is then transformed into the final ratings matrix R. This process not only restores the data to its original dimension but also refines the resolution, providing a high-fidelity reconstruction of the input.

These deep learning operations are crucial as they enable the CAE to transform the abstracted features back into a tangible output, which in our case, is the ratings matrix used for generating recommendations. To ensure the effectiveness of this process, the model employs backpropagation during training, optimizing the neural network to reduce prediction errors and enhance the accuracy of the final recommendation output.

In conclusion, our Convolutional Autoencoder model utilizes convolutional layers in the encoder to capture deep relationships within the user-item data, and deconvolution layers in the decoder to reconstruct and predict accurate ratings. This method ensures that our model effectively bridges the gap between deep learning feature extraction and practical recommendation

system applications. The experimental section will further explore the outcomes and effectiveness of these processes.

3.3 Experimental Evaluation

3.3.1 Experimental Framework

In our study, we implement and assess our convolutional autoencoder-based recommendation system using two distinct datasets. We benchmark our model's performance against other established deep learning approaches such as GAP [24], DNNRec [17], ExtKNNCF [26], URP-CF-SIM [27], DCN-M [38], and ECAE [28].

The comparison models are evaluated using the same datasets and metrics as CAERS, ensuring a fair and consistent assessment. Performance results for these models are derived from published research studies, with the cited sources providing the necessary context and validation for their metrics. This approach not only maintains transparency but also allows for a direct and meaningful comparison of CAERS with current state-of-the-art methods. Detailed explanations of the experimental setup, including dataset characteristics and evaluation metrics, are provided in Chapter 3. This standardized approach is consistently applied throughout the other chapters to ensure that all results are based on a uniform evaluation framework.

Our architecture employs 3×3 two-dimensional kernels in both the convolutional and transposed convolutional layers, optimizing the extraction and transformation of data into predictive ratings. The encoder consists of two convolutional layers, while the decoder is composed of three deconvolution layers, both segments supported by two max-pooling layers each, as depicted in Figure 3.4. Our primary focus is on reducing the Mean Squared Error (MSE), adapted to account

for the zero entries in our dataset that represent unrated items by users, effectively masking these during training to avoid skewed learning outcomes.

To simulate the cold start problem, we introduce new users—who were not part of the training or validation datasets—into the test dataset. We separate the test data into three distinct segments, where we progressively add new users in increments of 10%, 20%, and 30% of the test set. This method allows us to evaluate the model's performance under different levels of information scarcity and assess its robustness when faced with varying degrees of new, unseen user data.

The optimization of our model is achieved using the Adaptive Moment Estimation (Adam) [16] with a learning rate of 0.01. Activation functions include Rectified Linear Unit (RELU) for the hidden layers and sigmoid for the output layer. Our training batches consist of 32 samples each, with a data split ratio of 90:10 between training and testing, and an additional 10% of the training set reserved for validation.

Performance metrics employed are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

3.3.2 Datasets

Our evaluation employs two versions of the widely-used MovieLens dataset [12], specifically the 100K and 1M datasets. These datasets are well-regarded in the research community for benchmarking recommendation algorithms due to their substantial size, diversity, and real-world user rating data. The 100K dataset includes 943 users and 1,632 movies, resulting in 100,000 ratings, while the 1M dataset is significantly larger, containing 6,040 users and 3,883 movies, with a total of 1 million ratings. Both datasets exhibit varying degrees of sparsity, with numerous missing entries, which we systematically exclude from the training and validation phases to ensure robust performance evaluation.

The choice of these datasets is justified because they provide a balanced mix of sparsity and data density, allowing us to effectively test the scalability and generalizability of our model across different dataset sizes. Additionally, the datasets contain diverse demographic information—such as user occupation, age, and gender—and detailed movie attributes, including titles and genres, which are vital for building a more personalized recommendation system. To further enhance the contextual richness of the data, we supplement the item information with external sources that provide comprehensive movie overviews. These textual details are processed using the BERT technique [7] to capture deep contextual semantics, ensuring that our model can leverage nuanced content features to improve recommendation accuracy.

Using these two datasets is sufficient because they represent a wide range of user behaviors and preferences in a real-world context. The combination of diverse user demographics, item attributes, and external content allows us to thoroughly evaluate the performance and adaptability of our model, making it generalizable to other domains beyond movie recommendations.

Adapting to Other Domains: To expand the application of our recommendation system to different domains, such as music, books, or retail products, the following steps are suggested:

- **Identify Domain-Specific Features:** Begin by identifying the key attributes that are crucial for making recommendations in the new domain. For instance, in a music recommendation system, important features could include not only genre, artist, release date, and user listening behaviors but also mood, tempo, popularity metrics, and engagement patterns (such as skips, repeats, and playlist additions). In a book recommendation context, relevant features might include the author, genre, publication year, language, narrative style, and user reading behavior (such as reading speed, completion rate, and reviews).

- **Establish a Data Collection Plan:** Compile detailed data that accurately represents each item in the new domain. For example, in our movie recommendation system, we enriched

the dataset with supplementary information from external sources like Wikipedia to enhance the contextual understanding. A similar strategy can be used in other domains to ensure that the model has enough data for precise predictions. An effective encoding technique should also be employed to capture the contextual details.

- **Train the Model on the New Domain:** With the gathered data, create a user-item interaction matrix and a corresponding rating matrix for the new domain. The rating matrix might contain some items without ratings, which should be carefully handled during the training phase to prevent inaccuracies—unrated items represented as zeros should not be mistaken for actual ratings. Further details on this process can be found in the experimental setup section **??**.

3.3.3 Evaluation Metrics

In this thesis, we employ two primary evaluation metrics across various models and experiments: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). These metrics are crucial for quantifying the accuracy of prediction models within recommendation systems.

1. **Root Mean Square Error (RMSE)** quantifies the square root of the average of the squared differences between the predicted and actual values. It is particularly sensitive to large errors, making it highly effective in scenarios where it is crucial to avoid large deviations in predictions. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.8}$$

where y_i and $\hat{y}_i$ are the actual and predicted values, respectively, and n is the number of observations. RMSE is advantageous because it penalizes larger errors more significantly, ensuring that models aim to minimize substantial deviations.

2. **Mean Absolute Error (MAE)** measures the average magnitude of the errors in a set of predictions, without considering their direction. It is less sensitive to outliers than RMSE and provides a direct measure of average error magnitude:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.9}$$

MAE is beneficial for understanding the typical size of errors, providing a straightforward interpretation of model accuracy.

RMSE and MAE are chosen for their complementary characteristics in evaluating model performance. RMSE's sensitivity to large errors makes it suitable for identifying models that might produce significant prediction deviations, crucial in applications where such deviations are undesirable. Conversely, MAE offers a straightforward interpretation of average error magnitude, making it useful for understanding overall prediction accuracy in a simpler, more intuitive manner.

Both metrics have been extensively utilized in recommendation system evaluations, providing a balanced view of model performance. RMSE's focus on large errors ensures that the model penalizes significant inaccuracies, while MAE offers a clear picture of the average prediction error, facilitating a comprehensive evaluation of recommendation algorithms [10].

3.3.4 Results

The experimental results, presented in Table 3.1, demonstrate the effectiveness of our CAERS model in comparison to other established models across two datasets. On the MovieLens 100K dataset, CAERS achieved an RMSE of 0.9218 and an MAE of 0.7248, surpassing the second-best model, URP-CF-SIM, by 1.20% and 1.25% for RMSE and MAE respectively. On the MovieLens

1M dataset, CAERS showed a slight underperformance in RMSE by 0.31% compared to URP-CF-SIM but improved in MAE by 0.16%. These metrics are visually represented in Figures 3.5 and 3.7 for MovieLens 100K, and Figures 3.6 and 3.8 for MovieLens 1M.

Qualitative Analysis The results in Table 3.1 shows that the CAERS model outperforms several established models on the MovieLens 100K dataset, achieving the lowest RMSE (0.9218) and MAE (0.7248). This represents an improvement over the second-best model, URP-CF-SIM, with reductions of 1.20% in RMSE and 1.25% in MAE. These results suggest that CAERS's architecture is particularly effective at capturing intricate user-item interactions, leading to more accurate predictions.

For the MovieLens 1M dataset, CAERS demonstrates robust performance by maintaining the lowest MAE (0.6889), which is 0.16% better than URP-CF-SIM. Although CAERS slightly underperforms in RMSE by 0.31% compared to URP-CF-SIM, this minor difference is outweighed by its superior MAE performance, indicating better handling of absolute errors. This suggests that CAERS effectively balances prediction accuracy and stability, even when dealing with a larger, more complex dataset.

The results across both datasets highlight CAERS's ability to generalize well to different data scales, maintaining a balance between precision (RMSE) and reliability (MAE). The model's strengths are rooted in its convolutional autoencoder, which is adept at learning deep, nuanced representations of user and item features. By doing so, it reduces both types of errors more effectively than traditional models, making CAERS a versatile solution for real-world recommendation systems.

Overall, the findings confirm that CAERS is not only competitive but often superior to other models in terms of recommendation accuracy. Its capability to minimize errors across different datasets makes it particularly valuable for applications where reliable, high-quality predictions

are essential for enhancing user experience.

Dataset	Model	RMSE	MAE
MovieLens 100K	**CAERS**	**0.9218**	**0.7248**
	URP-CF-SIM	0.933	0.734
	GAP	0.9379	0.7343
	DCN-M	0.9552	0.7573
	DNNRec	0.9546	0.7532
	ECAE	0.9513	0.7566
	ExtKNNCF	1.037	0.810
MovieLens 1M	**CAERS**	**0.8817**	**0.6889**
	URP-CF-SIM	0.879	0.690
	GAP	0.8989	0.7048
	DCN-M	0.9220	0.7379
	DNNRec	0.9076	0.7122
	ECAE	0.9057	0.71
	ExtKNNCF	0.853	0.764

Table 3.1 Final Experimental Results of different models on MovieLens datasets

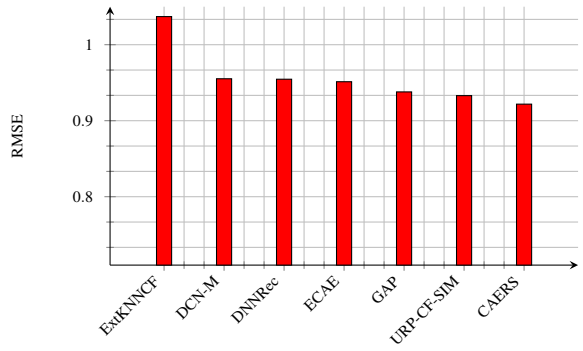

Figure 3.5: RMSE for MovieLens 100K

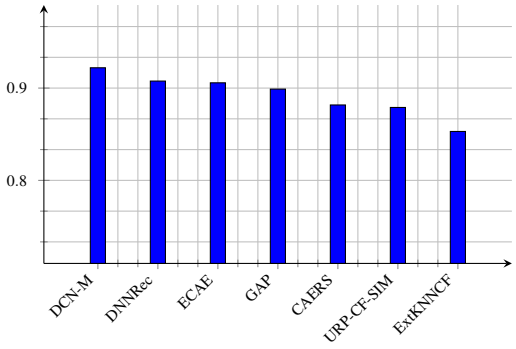

Figure 3.6: RMSE for MovieLens 1M

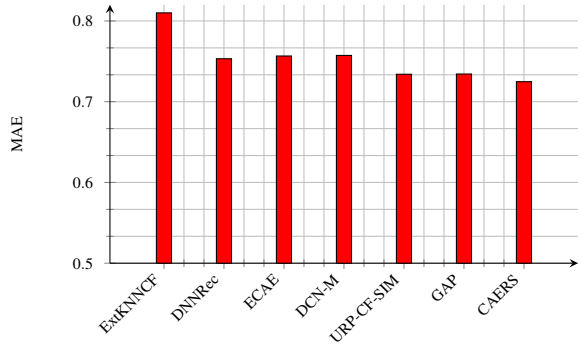

Figure 3.7: MAE for MovieLens 100K

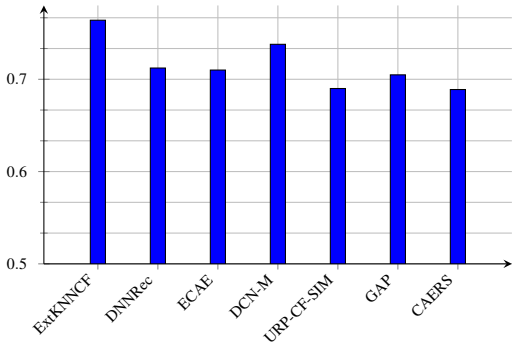

Figure 3.8: MAE for MovieLens 1M

Table 3.2 showcases the effectiveness of our CAERS model in tackling the cold start issue by incorporating new users at different increments (10%, 20%, 30%). The model demonstrated improvements in prediction accuracy of 1.62%, 2.26%, and 2.03% respectively for these groups. These metrics affirm the consistent superiority of our CAERS model over other leading deep learning models in cold start conditions. Overall, our model achieved an average error reduction of about 1.97% compared to the next best performing system.

Figure 3.9 visually illustrates how our model surpasses the GAP model and others, particularly in accurately predicting the preferences of new users who have no previous interactions

within the system.

This performance indicates that our CAERS model effectively forms significant connections between user characteristics and item content, independently of historical interaction data such as reviews or likes. This capability is pivotal for systems needing to quickly adapt to new user data without extensive historical inputs.

Model	**10%**	**20%**	**30%**
CAERS	1.008	1.0144	1.0173
GAP	1.0248	1.0377	1.0383
URP-CF-SIM	1.0410	1.0463	1.0498
DCN-M	1.0455	1.0576	1.0588
DNNRec	1.1077	1.1153	1.1413
ExtKNNCF	1.1209	1.2485	1.2505

Table 3.2: Prediction scores of new users

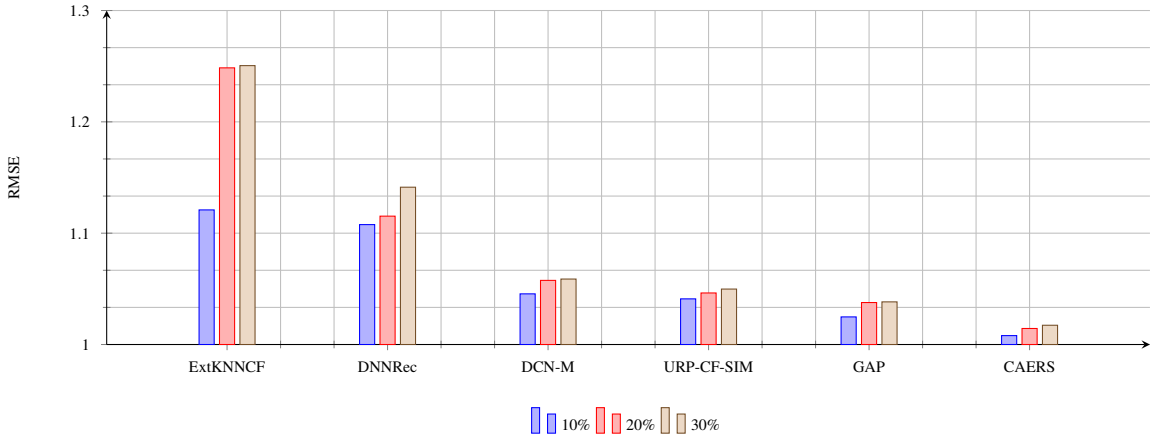

Figure 3.9: Prediction RMSE of New Users

3.4 Summary

In this chapter, we focused on the Convolutional Autoencoder recommendation system (CAERS), which utilizes advanced deep learning techniques to enhance the analysis of user-item interactions in recommendation systems. We explored the architecture of the CAE, which comprises an encoder that compresses input data to a latent space and a decoder that attempts to reconstruct the input data into a usable output—in this case, user ratings. This approach allows for sophisticated handling of the complexities inherent in recommendation scenarios, particularly beneficial for addressing the cold start problem where no previous user-item interaction data is available.

The chapter detailed the embedding processes used to handle both continuous and categorical variables within our datasets. By employing embedding techniques and a multi-layer perceptron, the system efficiently processes and interprets the data, capturing deeper contextual meanings that improve the accuracy of its predictions.

A significant part of the discussion highlighted how the CAE recommendation system learns to discern intricate relationships within the data, enabling it to offer precise recommendations that are specifically tailored to user preferences. Moreover, the system's capability to operate effectively without historical interaction data marks a significant stride toward solving the cold start problem in recommendation systems.

Overall, the chapter underscored the innovative application of CAE in recommendation systems, demonstrating its potential to significantly enhance how these systems understand and predict user preferences, thereby paving the way for more personalized and effective recommendation strategies in digital platforms.

Chapter 4

CAERS-CF: Hybridizing CAERS with Collaborative Filtering

4.1 Introduction

Building on the foundation laid by the Convolutional Autoencoder Recommendation System (CAERS), this section introduces the methodology behind the CAERS-CF hybrid model, which integrates the strengths of CAERS with model-based collaborative filtering (CF) techniques. This hybridization aims to enhance the recommendation system's accuracy and reliability by combining content-based predictions with behavior-based predictions.

4.1.1 Integration of Collaborative Filtering

Collaborative Filtering (CF) and Matrix Factorization (MF) are key techniques in building sophisticated recommendation systems. CF leverages user behavior to make predictions and rec-

ommendations, operating primarily through two approaches: memory-based and model-based. The model-based approach, including techniques like matrix factorization (MF), is particularly notable for its efficiency in managing large datasets and its proficiency in generating accurate recommendations amidst sparse data [42]. This is crucial in scenarios where user interactions with items are infrequent or incomplete, which is common in many real-world applications.

Matrix Factorization (MF) is a core algorithm underpinning model-based CF. It functions by decomposing the large user-item interaction matrix into two smaller, lower-dimensional latent feature matrices. One matrix represents the latent features of users, encapsulating user preferences, while the other represents item attributes. This decomposition helps in capturing the underlying factors driving user preferences and item characteristics, enabling the prediction of missing entries in the interaction matrix. These predictions correspond to potential user ratings or preferences, thus facilitating personalized recommendations.

The mathematical robustness of MF stems from its ability to reduce the dimensionality of the dataset while preserving its structural complexities. This is achieved through techniques such as Singular Value Decomposition (SVD) or Probabilistic Matrix Factorization, which optimize the latent features to best reconstruct the original matrix while minimizing overfitting [47].

In our methodology, we employ the matrix factorization technique, inspired by its robust mathematical framework and its proven effectiveness in the domain of recommendation systems. Unlike content-driven interaction matrices, such as those used in CAERS, the interaction matrix in CF is typically populated with direct user ratings. This provides a clearer, direct measure of user preferences, making it ideal for MF applications.

The matrix factorization process and its impact on recommendation systems are illustrated in Figure 4.1, highlighting how these latent features interplay to predict user-item interactions effectively. This visualization helps in understanding how MF can predict unseen ratings by

users, thus tailoring recommendations to individual tastes and preferences.

The Matrix Factorization model is succinctly expressed by the following formula:

$$R \approx EG^T \tag{4.1}$$

In this equation, R denotes the actual ratings matrix observed in the data. The matrices E and G represent the user and item feature matrices, respectively. Their product aims to approximate the true ratings matrix R. The effectiveness of this approximation is crucial as it directly influences the model's ability to predict user preferences accurately. To optimize E and G for the best approximation of R, algorithms such as stochastic gradient descent are employed [11]. These optimization techniques iteratively adjust the values in E and G to minimize the discrepancies between the predicted ratings and the actual ratings, thereby refining the model's predictions and reducing error.

4.1.2 Combining CAERS and CF

Once the separate predictions from CAERS and CF are obtained, the next crucial step in the hybrid recommendation system involves strategically merging these two distinct sets of outputs. This integration is achieved through the application of a linear regression model, which is specifically chosen for its simplicity and effectiveness in balancing the contributions of both models. The linear regression model is configured to dynamically adjust and find the optimal balance between the CAERS and CF predictions, ensuring that each model's strengths are appropriately weighted.

The primary goal of using linear regression here is to minimize the overall prediction error by combining the deep content analysis of CAERS with the behavioral insights of CF. This

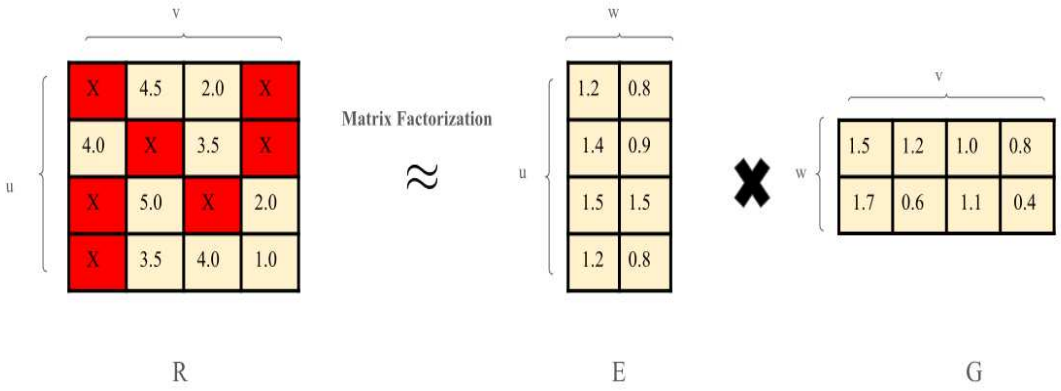

Figure 4.1 Matrix factorization process

method is favored over more complex algorithms because it avoids unnecessary computational complexity while providing clear, interpretable results. By leveraging the unique strengths of both systems in a straightforward manner, linear regression enhances the accuracy and relevance of the final recommendations, resulting in a more robust and reliable output.

The methodology for combining these predictions is mathematically represented by the following linear combination equation:

$$\hat{R} = W_0 + W_1 \cdot \hat{R}_1 + W_2 \cdot \hat{R}_2 \tag{4.2}$$

In this equation, $\hat{R}_1$ and $\hat{R}_2$ denote the predictions outputted by the CAERS and CF models, respectively. W_0, W_1, and W_2 are the weights allocated by the linear regression to harmonize the contributions of each model's predictions. $\hat{R}$ represents the final predicted ratings matrix, which

embodies the amalgamated recommendations. The role of the linear regression in this context is to meticulously optimize these weights to ensure that the combined model fits the observed data as accurately as possible, which is quantified using the mean squared error loss function:

$$J(W) = \frac{1}{n}\sum_{i=1}^{n}(R_i - \hat{R}_i)^2 \tag{4.3}$$

Here, R_i are the actual ratings given by users, $\hat{R}_i$ are the predicted ratings by the hybrid model, and n represents the total number of observations or data points in our dataset. The linear regression model adjusts the weights W_0, W_1, and W_2 iteratively using a gradient descent technique, which helps in refining these weights incrementally to minimize the error in predictions:

$$W_l \leftarrow W_l - \alpha\frac{\partial J(W)}{\partial W_l} \tag{4.4}$$

In the above equation, α signifies the learning rate, a crucial hyperparameter that controls the rate at which the model learns. It ensures that each step of the weight adjustment during the gradient descent is sized appropriately, neither too large to overshoot the minimum nor too small to delay convergence.

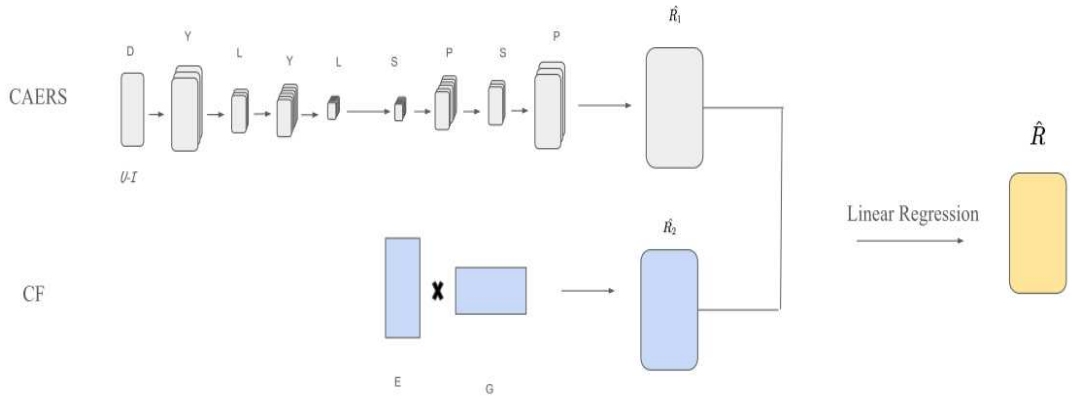

Figure 4.2 The proposed hybrid recommendation system CAERS-CF

This refined integration method highlights the synergetic potential of combining deep learning analysis and traditional predictive modeling to form a superior recommendation system that addresses complex user-item interaction scenarios with enhanced precision.

4.1.3 Expected Outcomes

The CAERS-CF methodology is designed to capitalize on the detailed content analysis of CAERS and the broad behavioral insights of CF. By integrating these approaches, CAERS-CF is expected to deliver superior performance, particularly in scenarios characterized by new users or sparse data, where traditional systems might struggle. The hybrid model not only aims to address the cold start problem more effectively but also enhances the personalization and accuracy of recommendations across diverse user segments.

In conclusion, the CAERS-CF model represents a sophisticated blend of deep learning and traditional recommendation techniques, promising advancements in both theory and application within the domain of recommendation systems.

4.2 Experimental Evaluation

This section of the thesis explores the performance of our hybrid recommendation system (CAERS-CF), which integrates the strengths of Collaborative Filtering (CF) and CAERS. The analysis focuses on comparing CAERS-CF not only against its constituent components—CAERS and CF—but also against other state-of-the-art recommendation systems. This approach highlights the synergistic enhancements and the competitive edge provided by the hybrid model, emphasizing its effectiveness in a broader context within the field.

4.2.1 Experimental Framework

In this section, we outline the experimental setup used to evaluate our hybrid recommendation system, CAERS-CF, using two datasets. CAERS-CF is assessed not only against its individual components—CAERS and CF—but also compared with leading hybrid deep learning methods such as GAP [24], DNNRec [17], ExtKNNCF [26], URP-CF-SIM [27], DCN-M [38], and ECAE [28].

The collaborative filtering component uses standard MSE for error calculation, with optimization through Stochastic Gradient Descent (SGD). This approach iteratively adjusts model parameters using randomly selected training data subsets, maintaining a steady learning rate of 0.01.

Our training regimen processes data in batches of 32 and follows a split of 90:10 for training and testing, respectively, with an additional 10% reserved for validation. We evaluate performance using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), metrics chosen to demonstrate the superior efficacy of CAERS-CF. All models undergo hyperparameter tuning to optimize performance further.

4.2.2 Result

Our evaluation, as detailed in Table 4.1, quantifies the performance improvements of CAERS-CF over its foundational elements using the RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) metrics. For the MovieLens 100K dataset, the hybrid model shows significant reductions in RMSE and MAE, with decreases of 6.75% and 8.03% over CF, and 4.52% and 5.46% over CAERS, respectively. Specifically, the RMSE and MAE values are recorded at 0.8801 and 0.6852 for CAERS-CF, compared to CF's 0.9439 and 0.7451, and CAERS's 0.9218 and 0.7248. In the context of the MovieLens 1M dataset, the performance advantage of CAERS-CF continues, with reductions of 5.21% and 6.10% in RMSE and 4.75% and 5.06% in MAE over CF and CAERS, respectively. The RMSE and MAE for CAERS-CF are 0.8398 and 0.6540.

It is important to recognize the inherent characteristics of deep learning models, such as their dependency on substantial data volumes. As the data volume increases, the effectiveness of deep learning models generally improves, reflecting the capacity of such models to extract and learn from complex patterns in larger datasets. This principle is clearly observed in the performance of CAERS-CF, where the larger MovieLens 1M dataset allows for a more pronounced improvement in recommendation accuracy. The corresponding bar charts (Figures 4.3, 4.4, 4.5, and 4.6) visually underscore the superior performance of CAERS-CF, affirming that with an increase in dataset size, deep learning components like those in CAERS-CF can significantly enhance

the quality of recommendations. This analysis not only confirms the efficacy of integrating deep learning with collaborative filtering but also highlights the potential of CAERS-CF to scale effectively with increased data, making it a robust solution for diverse and expansive recommendation environments.

Dataset	Model	RMSE	MAE
MovieLens 100K	CAERS-CF	0.8801	0.6852
	CAERS	0.9218	0.7248
	CF	0.9439	0.7451
MovieLens 1M	CAERS-CF	0.8398	0.6540
	CAERS	0.8817	0.6889
	CF	0.8860	0.6965

Table 4.1 Experimental results, comparing the hybrid model with its constituent models

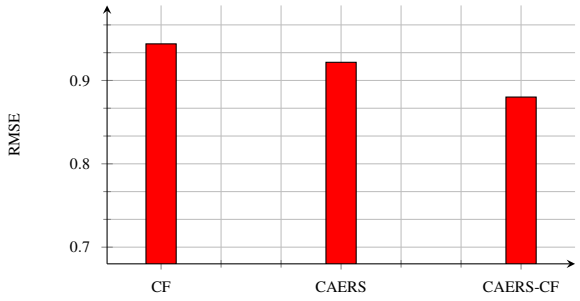

Figure 4.3 RMSE for MovieLens 100K

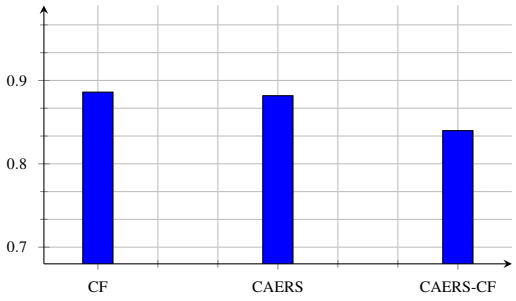

Figure 4.4 RMSE for MovieLens 1M

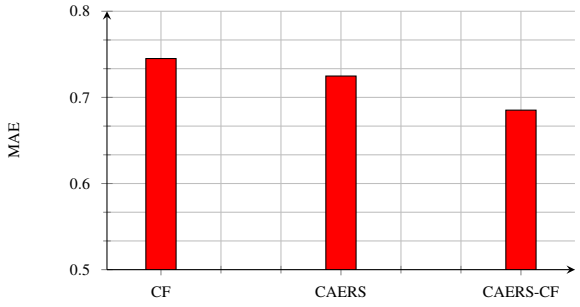

Figure 4.5 MAE for MovieLens 100K

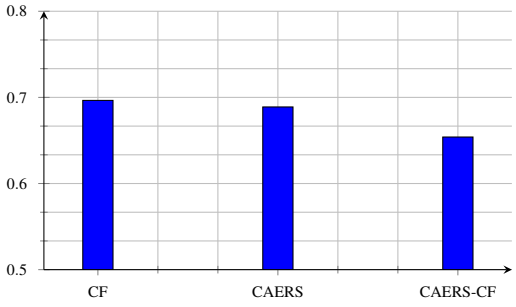

Figure 4.6 MAE for MovieLens 1M

In Table 4.2, we present a detailed comparison of our hybrid model, CAERS-CF, against

a variety of leading recommendation models using the MovieLens datasets. Additionally, this table highlights the performance of our deep learning-based recommendation system, CAERS, alongside other prominent models in this field.

The effectiveness of CAERS-CF is clearly demonstrated, not just in the tabulated data but also through visual representations in the bar charts for RMSE and MAE. These charts, shown in Figures 4.7 and 4.9 for the MovieLens 100K dataset and Figures 4.8 and 4.10 for the MovieLens 1M dataset, illustrate the superior performance of CAERS-CF.

On the MovieLens 100K dataset, CAERS-CF achieves an RMSE of 0.8801 and an MAE of 0.6852, showing improvements over various models: GAP (RMSE: 0.9379, MAE: 0.7343) with improvements of 6.16% and 6.69%, respectively; URP-CF-SIM (RMSE: 0.933, MAE: 0.734) with improvements of 5.67% and 6.65%, respectively; DCN-M (RMSE: 0.9552, MAE: 0.7573) with improvements of 7.86% and 9.52%, respectively; DNNRec (RMSE: 0.9546, MAE: 0.7532) with improvements of 7.80% and 9.03%, respectively; ECAE (RMSE: 0.9513, MAE: 0.7566) with improvements of 7.48% and 9.44%, respectively; ExtKNNCF (RMSE: 1.037, MAE: 0.810) with improvements of 15.13% and 15.41%, respectively.

For the MovieLens 1M dataset, CAERS-CF records an RMSE of 0.8398 and an MAE of 0.6540, surpassing several models: GAP (RMSE: 0.8989, MAE: 0.7048) with improvements of 6.57% and 7.21%, respectively; URP-CF-SIM (RMSE: 0.879, MAE: 0.690) with improvements of 4.46% and 5.22%, respectively; DCN-M (RMSE: 0.9220, MAE: 0.7379) with improvements of 8.92% and 11.37%; DNNRec (RMSE: 0.9076, MAE: 0.7122) with improvements of 7.47% and 8.17%, respectively; ECAE (RMSE: 0.9057, MAE: 0.71) with improvements of 7.27% and 7.89%, respectively; ExtKNNCF (RMSE: 0.853, MAE: 0.764) with improvements of 1.55% and 14.40%, respectively.

Additionally, Table 4.2 and the associated figures visually highlight the robust performance

of CAERS-CF, confirming its effectiveness relative to other contemporary models. Notably, even CAERS alone shows performance enhancements, with a reduction of 1.71% in RMSE and 1.30% in MAE for the 100K dataset, and improvements of 1.91% in RMSE and 2.25% in MAE for the 1M dataset.

These results substantiate the significant advancements our hybrid recommendation system has achieved over other state-of-the-art models, with both our novel deep learning approach and its hybrid extension proving to be highly effective when compared to traditional models.

Dataset	**Model**	**RMSE**	**MAE**
MovieLens 100K	CAERS-CF	0.8801	0.6852
	CAERS	0.9218	0.7248
	GAP	0.9379	0.7343
	URP-CF-SIM	0.933	0.734
	DCN-M	0.9552	0.7573
	DNNRec	0.9546	0.7532
	ECAE	0.9513	0.7566
	ExtKNNCF	1.037	0.810
MovieLens 1M	CAERS-CF	0.8398	0.6540
	CAERS	0.8817	0.6889
	URP-CF-SIM	0.879	0.690
	GAP	0.8989	0.7048
	DCN-M	0.9220	0.7379
	DNNRec	0.9076	0.7122
	ECAE	0.9057	0.71
	ExtKNNCF	0.853	0.764

Table 4.2 Final Experimental Results of different models on MovieLens datasets

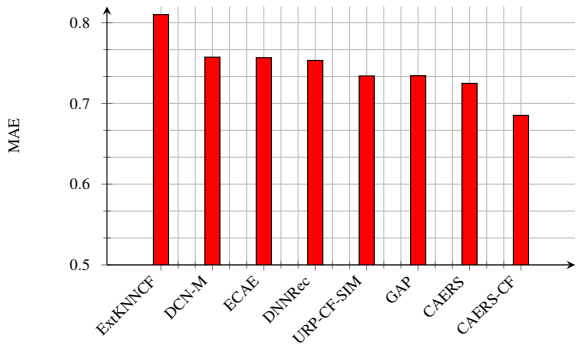

Figure 4.9: MAE for MovieLens 100K

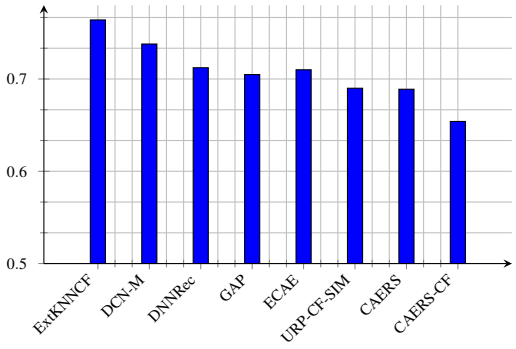

Figure 4.10: MAE for MovieLens 1M

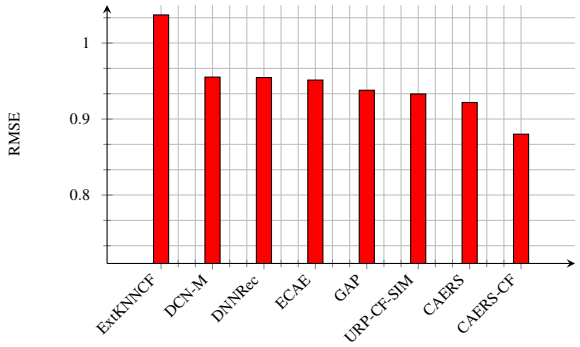

Figure 4.7: RMSE for MovieLens 100K

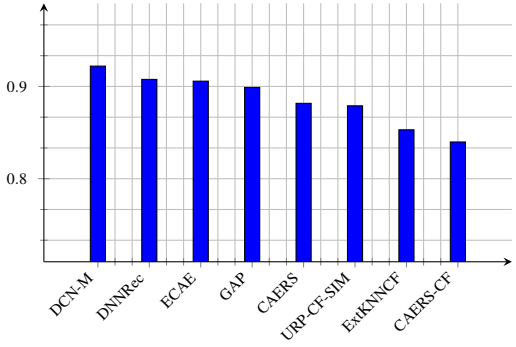

Figure 4.8: RMSE for MovieLens 1M

Weights: The final step, as previously discussed, involves determining the weights for each model through linear regression. The weights assigned to the outputs of the Collaborative Filtering (CF) and CAERS models are 0.289 and 0.811, respectively. This indicates that CAERS has a significantly greater role in our recommendation system compared to CF. In other words, CAERS exerts a stronger influence on the recommendations we generate than CF.

Qualitative Analysis: The CAERS-CF model demonstrates advancements in recommendation system performance through its hybrid approach, which combines Convolutional Autoen-

coders (CAERS) with Collaborative Filtering (CF). This integration leverages the strengths of both techniques, with CAERS providing detailed feature extraction from user and item data, while CF captures user-item interactions effectively. The result is a robust model that enhances recommendation accuracy by incorporating both content-based and collaborative insights.

The performance of CAERS-CF is impressive when compared to various contemporary models. On the MovieLens 100K dataset, CAERS-CF achieves an RMSE of 0.8801 and an MAE of 0.6852, indicating a high level of precision and minimal error in recommendations. This strong performance reflects the model's capability to handle complex recommendation scenarios. Similarly, for the MovieLens 1M dataset, CAERS-CF delivers an RMSE of 0.8398 and an MAE of 0.6540, showcasing its effectiveness in managing larger datasets and ensuring consistent recommendation quality.

The superiority of CAERS-CF is evident in its significant improvements over other advanced models such as GAP, URP-CF-SIM, DCN-M, DNNRec, and ECAE. CAERS-CF outperforms these models in both RMSE and MAE, underscoring its advanced capabilities and effectiveness. Additionally, the performance of CAERS alone also highlights its strength, with notable reductions in RMSE and MAE, further enhanced when combined with CF in CAERS-CF.

Visual representations in bar charts provide a clear illustration of CAERS-CF's effectiveness, reinforcing the quantitative results with compelling graphical evidence. Overall, CAERS-CF stands out as a highly effective recommendation system due to its hybrid approach, which combines CAERS and CF to deliver superior recommendation accuracy and performance.

4.3 Discussion on the Cold Start Problem

In the context of recommendation systems, the cold start problem poses a significant challenge, particularly when new users or items are introduced to the system. This issue arises because traditional collaborative filtering methods rely heavily on historical interaction data, which is lacking for new users or items, thus impeding the system's ability to provide accurate recommendations.

For the CAERS-CF model, while it demonstrates strong performance in leveraging both collaborative filtering and content-based features, it does not inherently address the cold start problem. The collaborative filtering component of CAERS-CF relies on historical user-item interactions, which can be insufficient or unavailable for new users or items. Consequently, this limitation affects the model's ability to generate accurate recommendations in scenarios where interaction data is sparse or nonexistent.

To mitigate the cold start problem, several strategies could be implemented to enhance CAERS-CF's robustness in such situations:

1. **Content-Based Switch:** One effective approach is to incorporate a switch mechanism within CAERS-CF that allows the system to toggle between collaborative-based outputs and content-based outputs. When the system detects a cold start scenario, it could automatically disable the collaborative filtering component and rely solely on the content-based recommendations provided by the CAERS architecture. This adjustment would enable the model to generate relevant recommendations based on item features and user profiles, even in the absence of interaction data.

2. **Hybrid Initialization:** Another approach is to employ hybrid initialization strategies that combine content-based methods with limited collaborative filtering data. For instance, when new items or users are introduced, the system could initially use content-based fea-

tures to provide preliminary recommendations and gradually incorporate collaborative filtering data as more interactions are collected over time. This hybrid approach helps bridge the gap between content-based and collaborative methods during the initial stages.

3. **User and Item Profiling:** Developing detailed user and item profiles through content-based features can also mitigate the cold start problem. By collecting rich contextual information and attributes for new users and items, the system can create a more comprehensive understanding of their preferences and characteristics. These profiles can then be used to generate initial recommendations and improve the model's performance as more interaction data becomes available.

4. **External Data Sources:** Leveraging external data sources and pre-trained models can further enhance CAERS-CF's ability to handle cold start scenarios. Integrating data from social media, reviews, or other external platforms can provide additional insights into new users or items, enriching the recommendation process and overcoming the limitations of sparse interaction data.

4.4 Summary

This thesis introduced CAERS-CF, a robust hybrid recommendation system that combines the strengths of two distinct recommendation methodologies to address a range of data types and scenarios. The first component of this hybrid system is the Convolutional Autoencoder Recommendation System (CAERS), a novel deep learning-based approach that uses a Convolutional Autoencoder (CAE) to process input data. CAERS effectively encodes and extracts essential content information from items and users, providing a deep understanding of their relationships, which is then used to generate precise rating predictions in the decoding phase.

The second component is a traditional model-based recommendation approach, Collaborative Filtering (CF), which utilizes the Singular Value Decomposition (SVD) technique. Unlike CAERS, which focuses on content data, CF leverages historical behavioral data to predict future user interactions and preferences. By analyzing past user behaviors, CF can predict how similar users will rate items, thereby enhancing the accuracy of the recommendations.

To fuse the outputs from these two systems, we employed a straightforward machine learning technique—linear regression. This method was used to ascertain the optimal weights for integrating the predictions from CAERS and CF, ensuring that each component contributes optimally based on its predictive strengths.

The performance of CAERS-CF was tested on two variants of the MovieLens dataset—100K and 1M. The results from these tests were effective, demonstrating significant enhancements in prediction accuracy as evidenced by reductions in both Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). These improvements highlight the efficacy of CAERS-CF in synthesizing the deep learning capabilities of CAERS with the established predictive power of traditional collaborative filtering.

Chapter 5

TriDeepRec: Innovating Hybrid Recommendations with Multi-Model Deep Learning Integration

5.1 Introduction

This chapter discusses the development of TriDeepRec, a hybrid recommendation system that integrates three deep learning approaches to improve the effectiveness of recommendations. TriDeepRec combines the capabilities of Convolutional Autoencoders, Neural Collaborative Filtering (NCF) [14], and Multilayer Perceptrons (MLP) [30], aiming to enhance the precision of recommendations.

TriDeepRec utilizes the content analysis provided by CAERS, the behavioral analysis from NCF, and the pattern recognition strength of MLP. This combination aims to tackle some common challenges in recommendation systems, such as dealing with sparse data and enhancing the

accuracy of predictions across various types of user interactions.

The goal of TriDeepRec is to optimize the predictive capabilities of each integrated model by blending their outputs through an advanced machine learning technique. This approach finely tunes the contributions from each component model to maximize overall recommendation quality. The result is a flexible and accurate system that excels at interpreting user preferences and item features to predict future interactions more effectively.

The efficacy of TriDeepRec has been evaluated against other leading models on benchmark datasets like MovieLens [28]. Its performance, particularly in reducing RMSE (Root Mean Square Error) and MAE (Mean Absolute Error), demonstrates an improvement over both its component (CAERS, NCF) and other deep learning approaches, showcasing its ability to provide personalized recommendations.

The remainder of this chapter will provide a comprehensive overview of TriDeepRec's methodology, its practical implementation, and the experimental outcomes that support its effectiveness. Additionally, the integration techniques and the synergistic operation of CAERS, NCF [14], and MLP [30] will be detailed, illustrating how TriDeepRec advances the field of hybrid recommendation systems.

5.2 TriDeepRec Methodology

We have previously demonstrated how the Convolutional Autoencoder Recommendation System (CAERS) and its hybrid extension, CAERS-CF, effectively utilize content information and past user behaviors to enhance recommendation accuracy. Building on this foundation, we now extend our exploration into Neural Collaborative Filtering (NCF). After a detailed examination of NCF, we will delve into how this approach can be integrated with a Multi-Layer Perceptron

(MLP). This integration aims to leverage the distinctive strengths of both models—NCF's user behavior analysis and MLP's advanced pattern recognition capabilities—to further elevate the precision and efficiency of our recommendation systems, thereby producing even more superior recommendations.

5.2.1 Neural Collaborative Filtering (NCF)

Neural Collaborative Filtering (NCF) is a sophisticated hybrid framework for recommendation systems that uniquely combines the predictive accuracy of Generalized Matrix Factorization (GMF) with the complex, non-linear modeling capabilities of a Multi-Layer Perceptron (MLP). This innovative combination enables NCF to adeptly capture a broad spectrum of user-item interaction patterns, ranging from straightforward linear correlations to intricate non-linear dynamics. The dual approach not only enhances the accuracy of recommendations but also offers a versatile model adaptable to diverse user behaviors and preferences (Figure 5.1).

NCF is particularly effective because it leverages the strengths of both deep learning architectures and traditional matrix-based approaches. Unlike CAERS, which primarily depends on content data, NCF utilizes past behavioral data to predict user preferences and item affinities, offering a robust foundation for generating reliable recommendations.

Generalized Matrix Factorization (GMF)

GMF is an advanced evolution of traditional matrix factorization techniques used extensively in recommendation systems. It focuses on deriving low-dimensional embeddings for users and items that efficiently capture the essence of their interactions through latent factors. These embeddings represent an abstraction of user preferences and item characteristics, providing a solid foundation for predicting interactions.

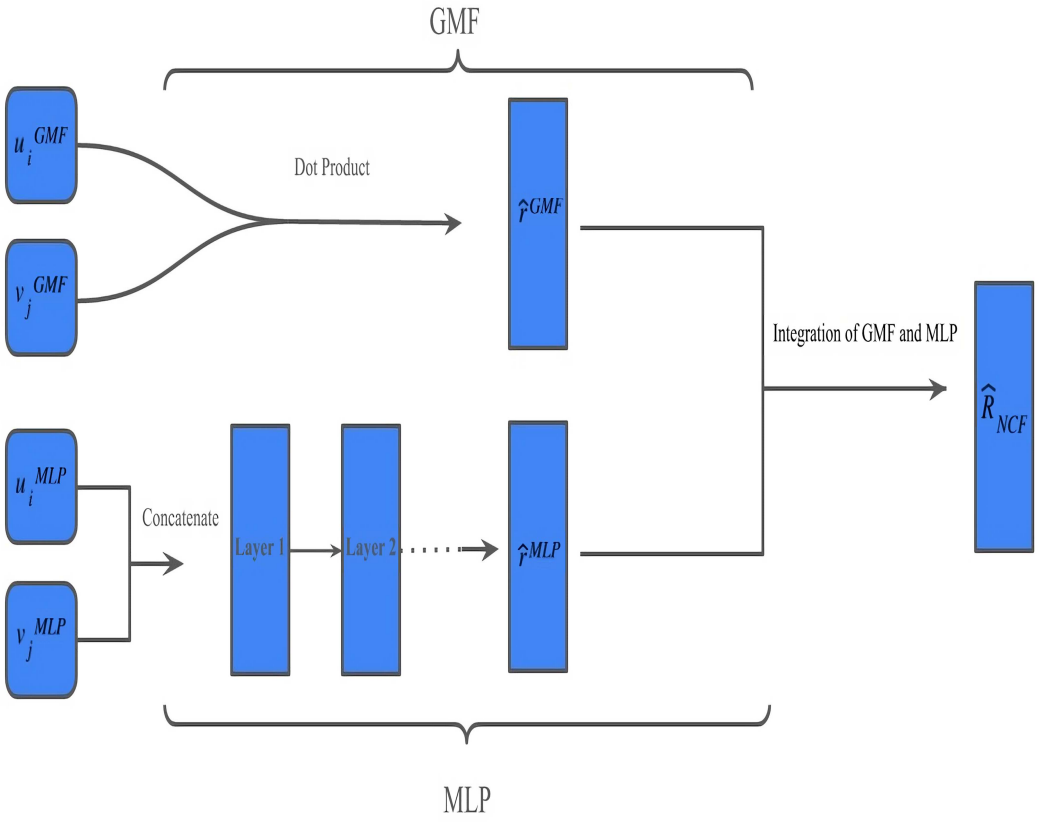

Figure 5.1: Illustration of the Neural Collaborative Filtering framework.

Embedding Process In the GMF model, users and items are represented by vectors in a latent space where each dimension corresponds to a latent feature that might influence their interactions. The embedding process for a user i and an item j is formalized as follows:

$$\mathbf{u}_i^{GMF} = \mathbf{E}_u^{GMF}(i), \tag{5.1}$$

$$\mathbf{v}_j^{GMF} = \mathbf{E}_v^{GMF}(j), \tag{5.2}$$

Here, $\mathbf{E}_u^{GMF}$ and $\mathbf{E}_v^{GMF}$ are the embedding functions that map users and items to their respective latent vectors within the GMF framework.

Interaction Prediction The interaction between a user and an item is predicted by the dot product of their respective latent vectors, embodying a linear relationship model:

$$\hat{r}_{ij}^{GMF} = \mathbf{u}_i^{GMF} \cdot \mathbf{v}_j^{GMF}. \tag{5.3}$$

This linear model is simple yet powerful, capable of capturing the essential characteristics that dictate user-item interactions (rating matrix).

Multi-Layer Perceptron (MLP)

MLP introduces a non-linear dimension to the recommendation process, using a series of layered neurons that enable the modeling of complex patterns. This capability is particularly useful in scenarios where interactions are influenced by factors that are not linearly separable or directly observable.

Embedding Process Similar to GMF, MLP uses an embedding process but tailors it to capture non-linear relationships:

$$\mathbf{u}_i^{MLP} = \mathbf{E}_u^{MLP}(i), \tag{5.4}$$

$$\mathbf{v}_j^{MLP} = \mathbf{E}_v^{MLP}(j), \tag{5.5}$$

where $\mathbf{E}_u^{MLP}$ and $\mathbf{E}_v^{MLP}$ represent the neural embedding functions for users and items, designed to facilitate non-linear processing.

Interaction Prediction The non-linear interaction scores are computed by processing the concatenated embeddings through a network of layers, each adding a level of complexity:

$$\hat{r}_{ij}^{MLP} = f(\mathbf{u}_i^{MLP} \| \mathbf{v}_j^{MLP}), \tag{5.6}$$

where f denotes a sequence of non-linear activation functions in the MLP, enhancing the model's ability to discern and learn from complex interaction patterns.

Integration of GMF and MLP

The final stage in NCF is the strategic integration of outputs from GMF and MLP, synthesizing them into a single unified prediction $\hat{R}_{NCF}$. This integration effectively combines the straightforward linear predictions from GMF with the intricate non-linear insights provided by MLP:

$$\hat{R}_{NCF} = \alpha \cdot \hat{r}_{ij}^{GMF} + (1-\alpha) \cdot \hat{r}_{ij}^{MLP}, \tag{5.7}$$

Here, α is a hyperparameter that balances the linear and non-linear contributions, optimizing the overall prediction accuracy. By fine-tuning α, NCF can be tailored to specific datasets and user behaviors, making it a highly adaptable and effective tool for generating personalized recommendations.

5.2.2 TriDeepRec

In our hybrid recommendation architecture, TriDeepRec, we effectively combine the strengths of the Convolutional Autoencoder Recommendation System (CAERS) with Neural Collaborative Filtering (NCF) using a Multilayer Perceptron (MLP). This combination aims to optimize the recommendation system by leveraging CAERS's proficiency in deep content analysis alongside NCF's insights into user-item interactions. The MLP acts as a critical integration point, processing the outputs from both CAERS and NCF to produce the final recommendation scores, illustrated in Figure 5.2.

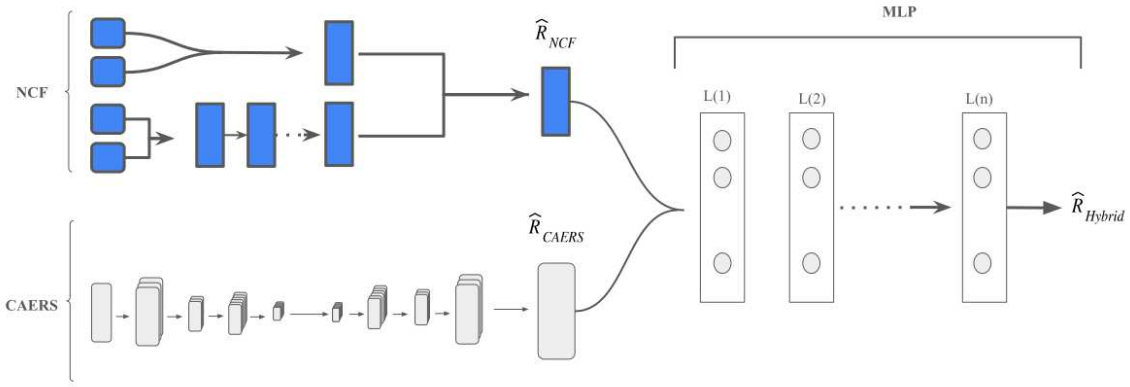

Figure 5.2: Diagram of the TriDeepRec hybrid recommendation system.

The process begins by extracting prediction ratings from both CAERS and NCF. These ratings, denoted as $\hat{R}_{CAERS}$ and $\hat{R}_{NCF}$, reflect the individual systems' assessments based on content similarity and past user interactions, respectively.

These prediction ratings are then fed into the MLP as a concatenated vector, forming a comprehensive dataset that encapsulates both content-based and interaction-based insights. The vec-

tor, denoted as $\mathbf{X}$, is defined as follows:

$$\mathbf{X} = [\hat{R}_{\text{CAERS}} \,||\, \hat{R}_{\text{NCF}}] \tag{5.8}$$

Within the MLP, this vector passes through multiple layers, each consisting of neurons that apply non-linear activation functions to capture complex relationships in the data. The transformation at each layer l can be expressed as:

$$\mathbf{X}^{(l)} = f^{(l)}(\mathbf{W}^{(l)}\mathbf{X}^{(l-1)} + \mathbf{b}^{(l)}) \tag{5.9}$$

The MLP culminates in an output layer that provides the final prediction $\hat{R}_{Hybrid}$, a scalar value that signifies the synthesized recommendation based on the integrated data:

$$\hat{R}_{Hybrid} = \mathbf{W}^{(L)}\mathbf{X}^{(L-1)} + \mathbf{b}^{(L)} \tag{5.10}$$

Here, L is the number of layers in the MLP. The system is optimized during training by minimizing a loss function, typically the Mean Squared Error (MSE), which measures the difference between the predicted ratings $\hat{R}_{Hybrid}$ and the actual user-item ratings:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(R_i - \hat{R}_{Hybrid,i})^2 \tag{5.11}$$

Training involves adjusting the MLP's weights $\mathbf{W}^{(l)}$ and biases $\mathbf{b}^{(l)}$ through backpropagation to minimize this error, ensuring that the hybrid model effectively combines insights from both CAERS and NCF to enhance recommendation performance.

5.3 Experimental Evaluation

This subsection details the setup and methodology employed in our research to evaluate the TriDeepRec hybrid recommendation model. We conduct experiments using two distinct datasets, comparing TriDeepRec not only with the CAERS and Neural Collaborative Filtering (NCF) models as standalone systems but also against advanced hybrid and deep learning-based methods including GAP [24], DNNRec [17], ExtKNNCF [26], URP-CF-SIM [27], DCN-M [38], and ECAE [28].

5.3.1 Experimental Framework

Neural Collaborative Filtering (NCF) Configuration The NCF module is structured with an embedding size of 64 to finely tune the user-item interaction nuances. It runs through six epochs with a learning rate of 0.001 and uses batch sizes of 32. To prevent overfitting, a dropout rate of 0.5 is applied. The architecture features layers configured with 64, 128, and 512 hidden units, optimized to detect complex interaction patterns without the use of negative sampling, focusing purely on recorded interactions. MSE serves as the loss function, chosen through rigorous hyperparameter tuning to ensure the model operates at its peak.

The integration of CAERS and NCF outputs is handled by a Multi-Layer Perceptron (MLP), which utilizes ReLU for activation and a very light regularization with an alpha value of 0.0001. The MLP includes one hidden layer with 32 neurons, a setup determined through hyperparameter tuning to optimally merge features from both models. This configuration aims to strike a balance between the model's learning capacity and complexity, enhancing the hybrid system's precision and relevance in predicting user preferences.

To gauge the effectiveness of TriDeepRec, we rely on Root Mean Squared Error (RMSE)

and Mean Absolute Error (MAE) as our primary metrics. These measures are instrumental in quantitatively assessing the accuracy of the model's predictions, allowing for a robust comparison against other models documented in the literature.

5.3.2 Result

This section delves into the evaluation of TriDeepRec, our advanced hybrid recommendation system, by benchmarking it against its components—CAERS and NCF—as well as other leading models in the field. We also revisit the performance of CAERS, particularly its effectiveness in addressing the cold start problem.

In our detailed comparison (Table 5.1), TriDeepRec's enhancements are quantified against CAERS and NCF using the MovieLens datasets. The analysis demonstrates TriDeepRec's improved accuracy in terms of both Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

On the MovieLens 100K dataset, TriDeepRec achieves an RMSE of 0.8845, which is a reduction of 4.05% from CAERS's RMSE of 0.9218 and a 5.50% improvement over NCF's RMSE of 0.9360. Similarly, TriDeepRec's MAE of 0.6963 surpasses that of CAERS at 0.7248 and NCF at 0.7417, marking improvements of 3.93% and 6.12%, respectively.

The evaluation extends to the MovieLens 1M dataset, where TriDeepRec maintains its lead by registering lower RMSE and MAE values than both CAERS and NCF. The RMSE improvement stands at 8.14% compared to CAERS and 7.90% compared to NCF. In terms of MAE, TriDeepRec shows a substantial enhancement, outperforming CAERS and NCF by approximately 9.16% and 9.13%, respectively, with a recorded MAE of 0.6258 for TriDeepRec versus 0.6889 for CAERS and 0.6887 for NCF.

Visual representations of these metrics (Figures 5.3, 5.4, 5.5, and 5.6) clearly illustrate TriDeepRec's dominance over the baseline models across both datasets, affirming its advanced capability to generate more accurate recommendations.

Dataset	Model	RMSE	MAE
MovieLens 100K	TriDeepRec	0.8845	0.6963
	CAERS	0.9218	0.7248
	NCF	0.9360	0.7417
MovieLens 1M	TriDeepRec	0.8099	0.6258
	CAERS	0.8817	0.6889
	NCF	0.8794	0.6887

Table 5.1 Comparative experimental outcomes between the hybrid model and its individual component models

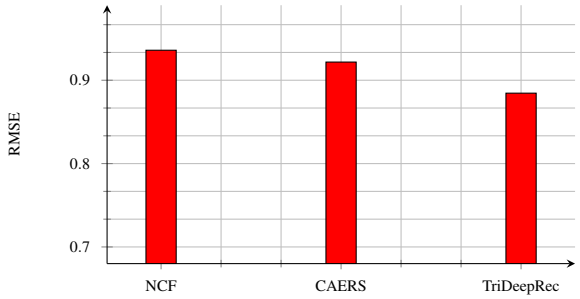

Figure 5.3 RMSE for MovieLens 100K

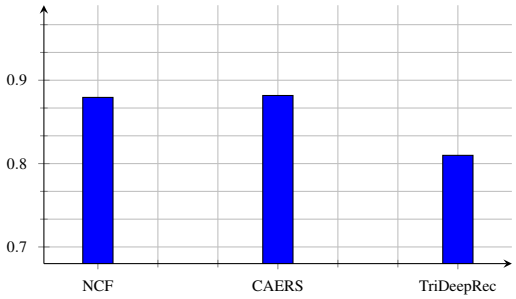

Figure 5.4 RMSE for MovieLens 1M

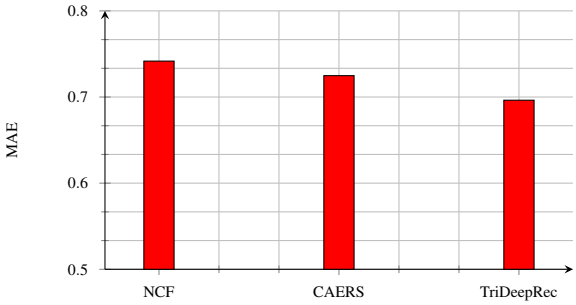

Figure 5.5 MAE for MovieLens 100K

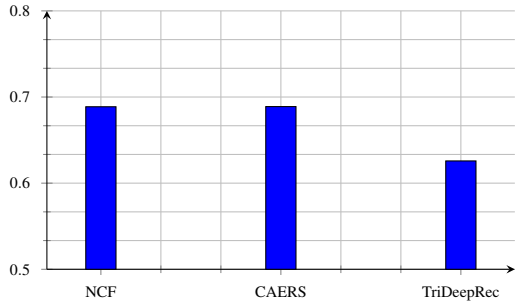

Figure 5.6 MAE for MovieLens 1M

Our comparative study, detailed in Table 5.2, evaluates the effectiveness of TriDeepRec by benchmarking it against several leading recommendation models, including GAP [24], DNNRec [17], ExtKNNCF [26], URP-CF-SIM [27], DCN-M [38], and ECAE [28].

On the MovieLens 100K dataset, TriDeepRec demonstrates superior performance, achieving the most favorable RMSE and MAE metrics. Specifically, TriDeepRec records an RMSE of 0.8845, which shows significant enhancements over GAP (0.9379), URP-CF-SIM (0.933), DCN-M (0.9552), DNNRec (0.9546), ECAE (0.9513), and ExtKNNCF (1.037), improving by 5.69%, 5.20%, 7.40%, 7.34%, 7.02%, and 14.71%, respectively. In terms of MAE, TriDeepRec attains a score of 0.6963, reflecting superior predictive accuracy with improvements of 5.17%, 5.14%, 8.05%, 7.55%, 7.97%, and 14.04% against the same models.

The analysis extends to the MovieLens 1M dataset where TriDeepRec continues to lead with the lowest recorded RMSE and MAE scores among the models evaluated. It achieves an RMSE of 0.8099 and an MAE of 0.6258, marking significant performance gains over GAP, DCN-M, DNNRec, ECAE, URP-CF-SIM, and ExtKNNCF with RMSE improvements of 9.90%, 12.16%, 10.76%, 10.58%, 7.86%, and 5.05%, respectively, and MAE improvements of 11.21%, 15.19%, 12.13%, 11.86%, 9.30%, and 18.09%, respectively. These results affirm TriDeepRec's robustness and reliability in generating accurate predictions across varying scales of data.

Visual representations of these comparative results are provided in Figures 5.7 and 5.9 for the MovieLens 100K dataset and Figures 5.8 and 5.10 for the MovieLens 1M dataset. These visuals clearly depict TriDeepRec's enhanced performance, confirming its efficacy in the field of recommendation systems.

Dataset	**Model**	**RMSE**	**MAE**
MovieLens 100K	TriDeepRec	0.8845	0.6963
	CAERS	0.9218	0.7248
	GAP	0.9379	0.7343
	URP-CF-SIM	0.933	0.734
	DCN-M	0.9552	0.7573
	DNNRec	0.9546	0.7532
	ECAE	0.9513	0.7566
	ExtKNNCF	1.037	0.810
MovieLens 1M	TriDeepRec	0.8099	0.6258
	CAERS	0.8817	0.6889
	URP-CF-SIM	0.879	0.690
	GAP	0.8989	0.7048
	DCN-M	0.9220	0.7379
	DNNRec	0.9076	0.7122
	ECAE	0.9057	0.71
	ExtKNNCF	0.853	0.764

Table 5.2 Comparative Outcomes of Various Models Across MovieLens Datasets

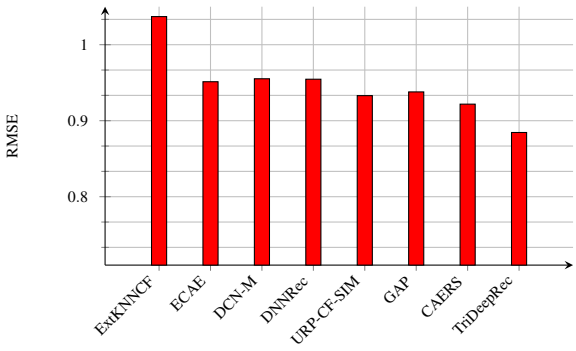

Figure 5.7 RMSE for MovieLens 100K

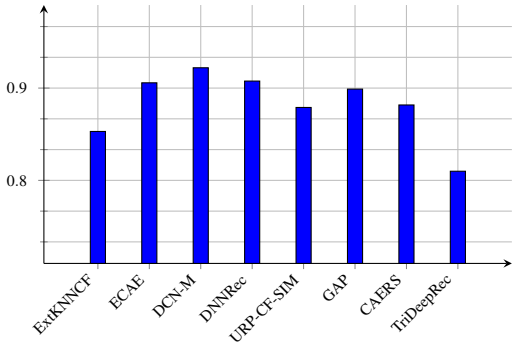

Figure 5.8 RMSE for MovieLens 1M

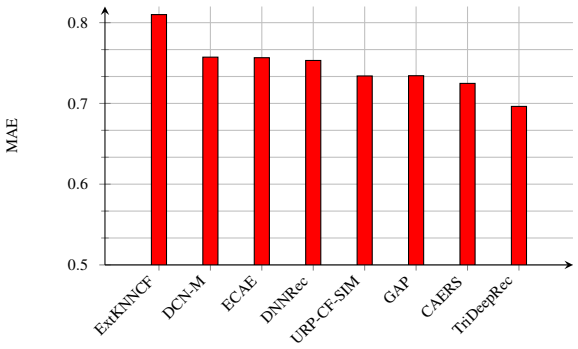

Figure 5.9 MAE for MovieLens 100K

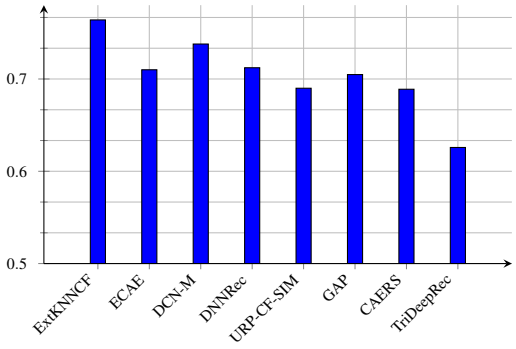

Figure 5.10 MAE for MovieLens 1M

Qualitative Analysis: While TriDeepRec demonstrates significant advancements in recommendation accuracy by leveraging a hybrid approach with three deep learning models, it is not always superior to CAERS-CF in all contexts. One notable reason for this is the risk of overfitting, which can arise from the complexity of the model.

TriDeepRec integrates three distinct deep learning models—content-based, behavior-based, and an additional layer for hybrid data processing. This multi-model architecture enhances the model's capacity to capture intricate patterns in the data. However, this increased complexity also comes with the risk of overfitting. Overfitting occurs when a model learns not only the underlying patterns but also the noise and specifics of the training data. As a result, while TriDeepRec might perform exceptionally well on training datasets, its generalization to new or unseen data can be less effective.

In contrast, CAERS-CF, which combines convolutional autoencoders and collaborative filtering, presents a more streamlined approach. Its architecture is designed to balance between capturing complex patterns and avoiding overfitting. By focusing on fewer components, CAERS-CF might better generalize to different datasets, particularly if it employs regularization techniques or simpler models that are less prone to overfitting.

Additionally, the hybrid nature of TriDeepRec, which combines three deep learning approaches, requires careful tuning and balancing of hyperparameters. The complexity introduced by these multiple layers can make it challenging to optimize the model effectively, potentially leading to suboptimal performance compared to CAERS-CF, which might benefit from a more focused and well-tuned approach.

Moreover, the increased computational complexity and longer training times associated with TriDeepRec can also be a drawback. The added complexity might lead to diminishing returns in performance improvements, particularly if the additional layers do not contribute significantly to capturing relevant patterns in the data.

While TriDeepRec offers an advanced and comprehensive approach to recommendation systems, its complexity and risk of overfitting can limit its performance relative to CAERS-CF. CAERS-CF's more focused architecture may provide better generalization and efficiency, making it a strong competitor despite the advanced design of TriDeepRec.

5.4 Discussion on Cold Start Problem

The TriDeepRec model, designed as a hybrid recommendation system, combines content-based and behavior-based data to enhance recommendation accuracy. However, it also faces challenges related to the cold start problem, which arises when new users or items lack sufficient interaction data.

To effectively address the cold start problem within the TriDeepRec framework, the following strategies can be considered:

1. **Enhanced Embeddings Initialization:** TriDeepRec can benefit from improved initialization strategies for embeddings. By using pre-trained embeddings or leveraging domain-

specific knowledge, the model can generate more meaningful initial representations for new users and items. For example, embeddings derived from auxiliary data sources, such as user demographics or item attributes, can provide a solid foundation for initial recommendations.

2. **Content-Based Reinforcement:** Incorporating a content-based reinforcement mechanism within TriDeepRec can enhance its performance during cold start scenarios. This approach involves boosting the influence of content-based features in the recommendation process when new users or items are introduced. By emphasizing content similarities, TriDeepRec can generate more relevant recommendations based on available attributes and features, even in the absence of sufficient interaction data.

3. **Cross-Domain Knowledge Transfer:** Utilizing cross-domain knowledge transfer can also be effective in handling cold start problems. TriDeepRec could integrate information from related domains or tasks where interaction data is more abundant. For instance, if the model is deployed in a new domain with limited data, it can transfer learned patterns and insights from similar domains to improve initial recommendations.

4. **Incremental Learning:** Implementing incremental learning techniques allows TriDeepRec to adapt continuously as new interaction data becomes available. By employing online learning or adaptive algorithms, the model can update its parameters in real-time and refine its recommendations based on new user-item interactions. This approach ensures that the model remains relevant and accurate as it accumulates more data over time.

5.5 Summary

This thesis introduces TriDeepRec, an innovative hybrid recommendation system that exemplifies the power of deep learning in enhancing recommendation technologies. TriDeepRec is a sophisticated integration of two distinct deep learning-based recommendation models—CAERS and NCF—with a Multilayer Perceptron (MLP). This combination harnesses the strengths of each model to significantly improve the accuracy and efficiency of recommendations.

The foundation of our approach is the Convolutional Autoencoder Recommendation System (CAERS), which effectively captures and utilizes content data to generate user-item ratings. By transforming content into actionable insights, CAERS sets the stage for a deeper understanding of user preferences based solely on item characteristics.

Building on this, we incorporate Neural Collaborative Filtering (NCF), a model that excels in analyzing user behavior data. NCF complements CAERS by adding a layer of intelligence that accounts for historical user interactions, thus enriching the recommendation process with a broader data perspective.

The synergy between CAERS and NCF is orchestrated through a Multilayer Perceptron, which acts as a fusion center. This MLP not only combines the outputs from both CAERS and NCF but also refines them through its network, resulting in highly accurate predictions. The effectiveness of TriDeepRec is empirically proven through rigorous testing on the MovieLens datasets, where it consistently outperforms existing models by showing marked improvements in both MAE and RMSE metrics.

Our results underscore the potential of integrating multiple deep learning techniques to address complex problems like recommendation accuracy and system efficiency. TriDeepRec not only advances the field of recommendation systems but also opens new avenues for future research, particularly in optimizing deep learning architectures for hybrid data processing.

In essence, the development of TriDeepRec represents a significant milestone in the pursuit of more intelligent, adaptive, and user-centric recommendation systems. The successes documented in this thesis highlight the transformative impact of deep learning innovations in overcoming traditional challenges in recommendation systems.

Chapter 6

Conclusion and Future Works

6.1 Introduction

This thesis has presented a comprehensive exploration of recommendation systems, with a focus on the integration of deep learning techniques to enhance prediction accuracy and user satisfaction. Throughout this work, we have developed and assessed various models, culminating in the introduction of the TriDeepRec model, which has demonstrated notable improvements over existing methods. This final chapter summarizes the key findings of our research, discusses the implications of these results, and outlines potential avenues for future work.

The conclusions drawn from this study not only highlight the strengths and weaknesses of current recommendation system technologies but also shed light on the significant impact that advanced machine learning techniques can have on this field. Additionally, this chapter aims to provide a roadmap for future research, suggesting several directions that can potentially lead to further enhancements in recommendation systems. We believe that the groundwork laid by this thesis will serve as a stepping stone for future innovations in the domain.

6.2 Conclusion

This thesis has delved into the development and detailed evaluation of TriDeepRec, a state-of-the-art hybrid recommendation system that effectively combines the strengths of deep learning models and sophisticated analytical methods to elevate the accuracy of recommendations. Our journey commenced with the creation of the Convolutional Autoencoder Recommendation System (CAERS), designed to adeptly predict user preferences using content data, thus addressing the prevalent cold start problem encountered in many recommendation systems.

CAERS leverages a convolutional autoencoder to intricately process complex content data into actionable insights, thereby enabling precise recommendations based on the distinct characteristics of items. This capability proves especially advantageous in promoting new or lesser-known items and ensures consistent performance even when user data is scarce.

Building on the capabilities of CAERS, we developed CAERS-CF, which amalgamates the predictive strengths of CAERS with the dynamic data-handling features of collaborative filtering. This integration significantly enhances the model's proficiency in utilizing historical interaction data, thereby augmenting its understanding and predictive accuracy regarding user preferences over time. By blending collaborative filtering, CAERS-CF taps into a broader spectrum of data, capturing more nuanced trends and preferences that might be overlooked by relying solely on content data.

Our most comprehensive model, TriDeepRec, merges the capabilities of CAERS and Neural Collaborative Filtering (NCF) with the processing power of a Multilayer Perceptron (MLP). This sophisticated integration harnesses both content and interaction data, refined further by the advanced analytical abilities of the MLP. Testing and validation have shown that TriDeepRec achieves notable improvements in recommendation accuracy, as evidenced by better performance metrics such as MAE and RMSE on the MovieLens datasets.

The successful validation of TriDeepRec underscores its superior performance compared to traditional systems and other deep learning-based models. By synthesizing the unique advantages of each component into a unified system, TriDeepRec adeptly addresses common issues such as scalability, data sparsity, and the harmonization of diverse data types.

This research enriches both the theoretical foundations and practical applications of recommendation systems. The innovative methodologies and integration strategies developed herein hold potential to significantly enhance commercial recommendation engines across various sectors, including e-commerce and media streaming.

In conclusion, this study not only demonstrates the effectiveness of integrating hybrid deep learning models to tackle traditional challenges in recommendation systems but also sets a pathway for future research. Potential improvements could come from further exploring user and item attributes, applying novel deep learning approaches, and integrating real-time data, which may collectively advance the capabilities of systems like TriDeepRec.

6.3 Future Works

Looking ahead, there are several promising directions to further enhance the performance of our hybrid recommendation systems. The exploration of integrating our Convolutional Autoencoder Recommendation System (CAERS) with other deep learning models offers an exciting opportunity to enrich our system's capabilities. By combining CAERS with models such as Generative Adversarial Networks (GANs) or reinforcement learning algorithms, we might develop more dynamic and responsive recommendation systems that adapt better to user feedback and evolving preferences.

Further efforts can also be directed toward improving how we utilize data for training our

models. There is potential to refine our data processing techniques to extract more detailed and significant insights from the user and item data. For instance, enhancing our feature extraction processes could lead to more detailed user profiles and item descriptions, which could help in crafting more personalized recommendations. Additionally, incorporating a broader range of user interaction data, such as click-through rates and browsing histories, might provide a fuller picture of user behavior and preferences, aiding in the fine-tuning of our recommendations.

Another area of future work could involve the use of large language models (LLMs) during the embedding phase of our system. LLMs could offer more sophisticated ways to encode and understand the nuances of content, potentially leading to better performance in capturing the subtleties of user-item relationships.

Moreover, we could explore developing a click-based method for our hybrid models. Such an approach would allow new users or items to engage with the system through simple interactions, like clicks, which could be used to temporarily prioritize content-based recommendations from CAERS when there is insufficient past behavior data. This method would ensure that new users or items are not disadvantaged, maintaining the system's effectiveness across varied user scenarios.

These directions not only aim to build on the successes of this thesis but also open up avenues for making recommendation systems more adaptive, inclusive, and accurate. As we continue to refine these technologies, the potential to significantly improve how users interact with and benefit from recommendation systems is considerable.

References

[1] 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*, 17(6):734–749, 2005.

[2] Xavier Amatriain and Justin Basilico. Recommender systems in industry: A netflix case study. In *Recommender systems handbook*, pages 385–419. Springer, 2015.

[3] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, pages 353–374, 2023.

[4] Greta Björklund, Magdalena Bohlin, Edvard Olander, Josef Jansson, Cicero Eduardo Walter, and Manuel Au-Yong-Oliveira. An exploratory study on the spotify recommender system. In *World Conference on Information Systems and Technologies*, pages 366–378. Springer, 2022.

[5] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *ArXiv Preprint ArXiv:1301.7363*, 2013.

[6] Aminu Da'u and Naomie Salim. Recommendation system based on deep learning methods: a systematic review and new directions. *Artificial Intelligence Review*, 53(4):2709–2748, 2020.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv Preprint ArXiv:1810.04805*, 2018.

[8] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

[9] Narges Ehsani, Farrokh Aminifar, and Hamed Mohsenian-Rad. Convolutional autoencoder anomaly detection and classification based on distribution pmu measurements. *IET Generation, Transmission & Distribution*, 16(14):2816–2828, 2022.

[10] Zeshan Fayyaz, Mahsa Ebrahimian, Dina Nawara, Ahmed Ibrahim, and Rasha Kashef. Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *applied sciences*, 10(21):7748, 2020.

[11] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77, 2011.

[12] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm Transactions on Interactive Intelligent Systems (tiis)*, 5(4):1–19, 2015.

[13] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. *ArXiv Preprint ArXiv:1808.03912*, 2018.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182, 2017.

[15] Mykola A Hodovychenko and Anastasiia A Gorbatenko. Recommender systems: models, challenges and opportunities. *Herald of Advanced Information Technology*, 4(6):308–319, 2023.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*, 2014.

[17] R Kiran, Pradeep Kumar, and Bharat Bhasker. Dnnrec: A novel deep learning based hybrid recommender system. *Expert Systems with Applications*, 144:113054, 2020.

[18] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[21] Soanpet Sree Lakshmi and T Adi Lakshmi. Recommendation systems: Issues and challenges. *International Journal of Computer Science and Information Technologies*, 5(4):5771–5772, 2014.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] Seungyeon Lee and Dohyun Kim. Deep learning based recommender system using cross convolutional filters. *Information Sciences*, 592:112–122, 2022.

[25] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32, 2015.

[26] Luong Vuong Nguyen, Quoc-Trinh Vo, and Tri-Hai Nguyen. Adaptive knn-based extended collaborative filtering recommendation services. *Big Data and Cognitive Computing*, 7(2):106, 2023.

[27] Sofia Nudrat, Hikmat Ullah Khan, Saqib Iqbal, Mian Muhammad Talha, Fawaz Khaled Alarfaj, Naif Almusallam, et al. Users' rating predictions using collaborating filtering based on users and items similarity measures. *Computational Intelligence and Neuroscience*, 2022, 2022.

[28] Yiteng Pan, Fazhi He, and Haiping Yu. A novel enhanced collaborative autoencoder with knowledge distillation for top-n recommender systems. *Neurocomputing*, 332:137–148, 2019.

[29] Yiteng Pan, Fazhi He, and Haiping Yu. A correlative denoising autoencoder to model social influence for top-n recommender system. *Frontiers of Computer science*, 14:1–13, 2020.

[30] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588, 2009.

[31] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *ArXiv Preprint ArXiv:1908.10084*, 2019.

[32] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Techniques, applications, and challenges. *Recommender systems handbook*, pages 1–35, 2021.

[33] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1):59, 2022.

[34] Marwa Sharaf, Ezz El-Din Hemdan, Ayman El-Sayed, and Nirmeen A El-Bahnasawy. A survey on recommendation systems for financial services. *Multimedia Tools and Applications*, 81(12):16761–16781, 2022.

[35] Ayush Singhal, Pradeep Sinha, and Rakesh Pant. Use of deep learning in modern recommendation system: A summary of recent works. *ArXiv Preprint ArXiv:1712.07525*, 2017.

[36] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009, 2009.

[37] I Nyoman Switrayana and Nur Ulfa Maulidevi. Collaborative convolutional autoencoder for scientific article recommendation. In *2022 9th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pages 96–101. IEEE, 2022.

[38] Ruoxi Wang, Rakesh Shivanna, Derek Z Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H Chi. Dcn-m: Improved deep & cross network for feature cross learning in web-scale learning to rank systems. *ArXiv Preprint ArXiv:2008.13535*, 2020.

[39] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 153–162, 2016.

[40] Xin Xin, Bo Chen, Xiangnan He, Dong Wang, Yue Ding, and Joemon M Jose. Cfm: Convolutional factorization machines for context-aware recommendation. In *IJCAI*, volume 19, pages 3926–3932, 2019.

[41] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, volume 17, pages 3203–3209. Melbourne, Australia, 2017.

[42] Baolin Yi, Xiaoxuan Shen, Hai Liu, Zhaoli Zhang, Wei Zhang, Sannyuya Liu, and Naixue Xiong. Deep matrix factorization with implicit feedback embedding for recommendation system. *IEEE Transactions on Industrial Informatics*, 15(8):4591–4601, 2019.

[43] Ruiping Yin, Kan Li, Guangquan Zhang, and Jie Lu. A deeper graph neural network for recommender systems. *Knowledge-Based Systems*, 185:105020, 2019.

[44] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[45] Su-Zhi Zhang, Peng-Hui Li, and Xiao-Ni Chen. Collaborative convolution autoencoder for recommendation systems. In *Proceedings of the 2019 8th International Conference on Networks, Communication and Computing*, pages 202–207, 2019.

[46] Yifei Zhang. A better autoencoder for image: Convolutional autoencoder. In *ICONIP17-DCEC. Available online: http://users. cecs. anu. edu. au/Tom. Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58. pdf (accessed on 23 March 2017)*, 2018.

[47] Yuan Zhang, Xueqing Lu, Yue Shi, and Doudou Zhang. Hybrid algorithm for item collaborative filtering based on matrix factorization. In *2023 4th Information Communication Technologies Conference (ICTC)*, pages 276–284. IEEE, 2023.

APPENDICES

This section provides the Python code used to implement the recommendation models discussed in the thesis, ensuring transparency and ease of replication. The appendices are organized as follows:

- **Appendix A** contains the code for the CAERS model, highlighting how it applies deep learning techniques to address the cold start problem.
- **Appendix B** includes the code for the CAERS-CF model, which combines deep learning with collaborative filtering to enhance prediction accuracy.
- **Appendix C** features the code for the TriDeepRec system, demonstrating the integration of multiple model predictions through a Multi-Layer Perceptron for improved recommendation accuracy.

These appendices offer a detailed look at the code behind each model, facilitating further development and experimentation.

Appendix A

Python Code for CAERS

The Python code for the CAERS model demonstrates the application of a Convolutional Autoencoder, featuring two convolutional layers in the encoder and three in the decoder, to extract meaningful information from the user-item matrix. It also includes the implementation of masked loss functions and metrics, providing a comprehensive view of the model's structure and performance evaluation techniques.

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D,
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split

# Data loading and preprocessing
dot = pd.read_pickle('DOT.pkl')  # Load dataset
```

```
labels = pd.read_pickle('LABELS.pkl')  # Load labels
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(dot, labels, ...)

# Custom loss function for evaluating model performance
def custom_loss(y_true, y_pred):
    # Apply a mask to ignore zero entries in the dataset
    mask = y_true > 0
    y_true_masked = tf.boolean_mask(y_true, mask)
    y_pred_masked = tf.boolean_mask(y_pred, mask)
    # Calculate Mean Squared Error on the masked data
    squared_errors = tf.square(y_true_masked - y_pred_masked)
    return tf.reduce_mean(squared_errors)

# RMSE metric for model evaluation
def RMSE(y_true, y_pred):
    mask = y_true > 0
    y_true_masked = tf.boolean_mask(y_true, mask)
    y_pred_masked = tf.boolean_mask(y_pred, mask)
    # Calculate Root Mean Squared Error on the masked data
    mse = tf.reduce_mean(tf.square(y_true_masked - y_pred_masked))
    return tf.sqrt(mse)

# MAE metric for model evaluation
def MAE(y_true, y_pred):
```

```
    mask = y_true > 0
    y_true_masked = tf.boolean_mask(y_true, mask)
    y_pred_masked = tf.boolean_mask(y_pred, mask)
    # Calculate Mean Absolute Error on the masked data
    return tf.reduce_mean(tf.abs(y_true_masked - y_pred_masked))

# 2D Convolutional Autoencoder Architecture
def convolutional_autoencoder(input_shape):
    # Encoder
    input_img = Input(shape=input_shape)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(input_img)
    x = MaxPooling2D((2, 2), padding='same')(x)
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
    encoded = MaxPooling2D((2, 2), padding='same')(x)
    # Decoder
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)
    x = UpSampling2D((2, 2))(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)
    return Model(input_img, decoded)

# Model creation and compilation
autoencoder_2d = convolutional_autoencoder((None, None, 1))
autoencoder_2d.compile(optimizer='adam', loss=custom_loss, metrics=[RMSE, MAE])
autoencoder_2d.summary()  # Display model architecture
```

Appendix B

Python Code for CAERS-CF

The provided code snippet demonstrates a method for enhancing recommendation accuracy by integrating predictions from two different models: a custom deep learning-based model (CAERS) and a matrix factorization model (SVD). Initially, the SVD model is trained on user-movie rating data to generate baseline predictions. Simultaneously, predictions from the pre-trained CAERS model are imported. These two sets of predictions are then merged and used as input features for a linear regression model, which is trained to find the optimal combination of both prediction sets, effectively leveraging the strengths of both models to improve the overall prediction accuracy.

```
import numpy as np
from sklearn.linear_model import LinearRegression
from surprise import Reader, Dataset, SVD
import pandas as pd

# Load the dataset
```

```
ratings = pd.read_csv('ratings.csv')
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(ratings[['user_id', 'movie_id', 'rating']], reader)

# Use SVD algorithm from Surprise package to predict ratings
svd = SVD()
trainingSet = data.build_full_trainset()
svd.fit(trainingSet)
predictions_svd = svd.test(trainingSet.build_testset())

# Assume 'caers_predictions.pkl' contains the CAERS model predictions
caers_predictions = pd.read_pickle('caers_predictions.pkl')

# Combine SVD predictions with CAERS predictions
combined_predictions = np.column_stack(
(caers_predictions, [pred.est for pred in predictions_svd]))

# True ratings
true_ratings = ratings['rating'].values

# Apply Linear Regression to find the optimal weights for combining predictions
model = LinearRegression()
model.fit(combined_predictions, true_ratings)

# Extract the coefficients (weights) from the model
```

```
weights = model.coef_
intercept = model.intercept_

# Calculate the combined weighted predictions
weighted_predictions = combined_predictions.dot(weights) + intercept

# Print model weights and intercept
print("Weights:", weights)
print("Intercept:", intercept)
```

Appendix C

Python Code for TriDeepRec

The script provided below illustrates the integration of two distinct recommendation system approaches: the collaborative filtering predictions from a CAERS model and the Neural Collaborative Filtering (NCF) predictions. After loading the previously saved CAERS predictions and setting up the NCF model, the script proceeds to train the NCF on user-item interaction data. Once both models have generated their respective predictions, these outputs are combined and fed into a Multi-Layer Perceptron (MLP). This MLP is trained on the combined dataset to learn the optimal way to integrate these predictions, ultimately aiming to enhance the predictive accuracy of the final recommendation system.

```
import numpy as np
import pandas as pd
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
from libreco.data import random_split, DatasetPure
from libreco.algorithms import NCF  # Neural Collaborative Filtering
```

```
# Load CAERS model predictions
caers_predictions = pd.read_pickle('caers_predictions.pkl')

# Load and prepare rating data
ratings = pd.read_csv('ratings.csv')
train_data, eval_data, test_data = random_split(ratings,..)
train_data, data_info = DatasetPure.build_trainset(train_data)

# Initialize and train NCF model
ncf = NCF(task="rating", data_info=data_info, embed_size=64,
n_epochs=6, lr=0.001, batch_size=32)
ncf.fit(train_data, verbose=2, eval_data=eval_data,
metrics=["rmse", "mae"])

# Predict ratings using the NCF model
ncf_predictions = ncf.predict(test_data)

# Combine NCF and CAERS predictions
combined_data = np.column_stack(
(ncf_predictions, caers_predictions))

# Apply a Multi-Layer Perceptron to the combined dataset
mlp = MLPRegressor(hidden_layer_sizes=(32, 64), max_iter=1000,
random_state=42, early_stopping=True)
```

```
mlp.fit(combined_data[:, :-1],
combined_data[:, -1])  # assuming last column is true ratings
```