Cross-Domain Recommender Systems in Cold-Start Problem

Sepehr Omidvar

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Masterin Computer Science

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

© Sepehr Omidvar, Ottawa, Canada, 2025

Examining Committee

The following served on the Examining Committee for this thesis.

External Examiner:	Junfeng Wen Assistant Professor, School of Computer Science University of Carleton
Internal Member(s):	Mehrdad Sabetzadeh Associate 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 today's digital ecosystem, recommendation systems are integral to shaping personalized user experiences across a wide range of applications, from e-commerce to entertainment to healthcare and education. Through the analysis of users' preferences and the prediction of future behavior, these systems reduce information overload and improve user satisfaction. While traditional recommendation models have become widely accepted, they still face significant challenges, such as handling sparse data, diverse user needs, and the dynamic nature of content preferences. To meet these limitations, innovative approaches are needed to ensure accurate, scalable, and personalized recommendations.

Among the potential solutions to these challenges is cross-domain recommendation, which makes use of knowledge from a source domain to improve recommendations in a target domain. This method is especially useful for overcoming cold-start situations, where few user or item interactions limit conventional models. By enabling knowledge transfer across domains, cross-domain recommendation systems provide a framework to overcome data sparsity and enhance prediction accuracy. The recent development of deep learning and transfer learning offers powerful tools for capturing complex user preferences and seamlessly transferring them from one domain to another. There is no doubt that designing effective architectures and strategies for such systems is not an easy task, as it requires careful consideration of domain-specific nuances, layer depth optimization, and the identification of transferable features.

In this thesis, we design a novel cross-domain recommendation system that utilizes personalized bridge functions and attention mechanisms to assist effective knowledge transfer between domains. By combining user embeddings with transferable characteristic encoding, the proposed framework captures user-specific preferences and dynamically adapts them to the target domain. A deep meta-network is employed to generate personalized bridges, allowing the system to adjust the unique behaviors of individual users. An in-depth evaluation of a real-world dataset demonstrates the superiority of this approach over traditional methods and outperforms the state-of-the-art model in addressing the cold-start

problem while maintaining high levels of accuracy.

Building upon this foundation, the system is further optimized through extensive analysis of neural network architecture, with a focus on layer depth and parameter tuning. By systematically evaluating the impact of different architectural configurations, this thesis identifies the optimal layer depth that balances model complexity with performance. These optimizations enable the model to achieve state-of-the-art performance in evaluation metrics like Mean Absolute Error and Root Mean Squared Error. This thesis contributes to the advancement of cross-domain recommender systems, providing a scalable and robust framework for addressing real-world challenges in personalized recommendation.

Acknowledgements

Throughout my academic journey, I have been fortunate to receive continuous support and guidance from many remarkable individuals. First and foremost, I want to express my deepest gratitude to my supervisor, Professor Thomas Tran, for his invaluable mentorship and encouragement throughout the course of my studies. His insightful advice and thoughtful guidance have greatly enriched my research and helped me grow both as a scholar and an individual. Under his guidance, I had the privilege of writing my first research paper, an experience that significantly shaped my understanding of academic rigor and collaboration. Professor Tran's dedication to his students and his ability to provide clarity during challenging moments have been a cornerstone of my academic success. I am truly thankful for the time and effort he has invested in my education.

I would also like to extend my heartfelt appreciation to my family, whose support has been the foundation of my achievements. My brother has always been a source of wisdom, offering a broader perspective and helping me find the right path forward. His guidance and belief in my potential have been invaluable in navigating the complexities of my studies. My mother has been my emotional anchor, consistently offering encouragement and energy that fueled my determination to work harder and persevere through challenges. I am eternally grateful for her unwavering love and support.

Finally, I want to honor the memory of my father, who, during his lifetime, was a source of strength and belief in my abilities. His confidence in my future success and his enduring support continue to inspire me to strive for excellence. Although he is no longer with us, his influence and encouragement remain a driving force in my life. To my family and my supervisor, I owe deep gratitude for shaping my journey and enabling me to reach this milestone.

Table of Contents

List of Tables

List of Figures

Abbreviations

BPR Bayesian Personalized Ranking 18, 20

CBF Content-based Filtering 3, 14, 15, 29, 70

CDR Cross-domain Recommendation 1, 6, 8, 9, 17, 26, 44, 47, 54, 64, 66, 68–72

CF Collaborative Filtering 3, 14–17, 26, 70

DCDCSR Deep Framework for Cross-domain and Cross-system 17–19, 23

DL Deep Learning 1, 9

DNN Deep Neural Network 8–10, 18, 20, 42, 50, 53, 65

DPTUPCDR Deep Personalized Transfer of User Preferences for Cross-domain Recommendation 2, 25, 27, 40, 47, 48, 69

EA Evolutionary Algorithm 49

ELU Exponential Linear Unit 10, 38, 42, 47, 52, 53, 57

EMCDR Embedding and Mapping for Cross-domain Recommendation 17, 18, 23, 28, 32, 37

GA Genetic Algorithm 49

GELU Gaussian Error Linear Unit 10

LLM Large Language Model 2

MAE Mean Absolute Error 35, 39, 40, 46–48, 57, 60, 69, 71

MF Matrix Factorization 15, 16, 18, 20, 21, 37

ML Machine Learning 9, 41

MLP Multi-layer Perceptron 18, 19

MMMF Maximum-margin Matrix Factorization 20

NLP Natural Language Processing 2, 11, 71

OneCycleLR One Cycle Learning Rate 28, 37, 41, 42, 47

PMF Probabilistic Matrix Factorization 20

PSO Particle Swarm Optimization 49

PTUPCDR Personalized Transfer of User Preferences for Cross-domain Recommendation 17, 18, 21, 28, 32, 37, 40

ReLU Rectified Linear Unit 10, 42, 43, 47, 52

RMSE Root Mean Squared Error 35, 39, 40, 46–48, 57, 60, 69, 71

SGD Stochastic Gradient Descent 10, 37, 53

SVD Singular Value Decomposition 15, 49

TL Transfer Learning 1, 8, 9, 12, 13, 17, 69

Chapter 1

Introduction

1.1 Overview

This thesis explores the use of deep learning and transfer learning to address key challenges in recommender systems, with a particular focus on Cross-domain Recommendation (CDR). CDR is especially valuable for solving the cold-start problem by transferring knowledge from a source domain with sufficient data to a target domain with limited user-item interactions. This approach improves the quality of recommendations in scenarios where there is little historical data available, enabling more effective personalization for users. The thesis is structured into five chapters:

Chapter 1: Introduction This chapter provides an overview of the thesis, outlining the motivations behind the research, the problem being addressed, and the key contributions. It also includes the publications that have resulted from this work.

Chapter 2: Background Information, Literature Review, and Related Work This chapter reviews existing research on recommender systems, Transfer Learning (TL), Deep Learning (DL), and CDR. It identifies limitations in current approaches, particularly in addressing the cold-start problem. The chapter justifies the need for CDR by highlight-

ing scenarios where traditional recommendation methods fall short or are not applicable. This sets the stage for the solution proposed in the next chapters.

Chapter 3: Tackling Cold-Start with Deep Personalized Transfer of User Preferences for Cross-domain Recommendation This chapter introduces the Deep Personalized Transfer of User Preferences for Cross-domain Recommendation (DPTUPCDR) framework, which employs personalized bridge functions to tackle the cold-start problem by transferring user preferences from a source domain to a target domain. The model demonstrates its effectiveness through empirical evaluation on multiple Amazon datasets, outperforming state-of-the-art methods by leveraging deep learning and meta-learning techniques.

Chapter 4: Layer Depth Matters: The Goldilocks Effect in Deep Cross-domain Recommendation This chapter extends the findings of Chapter 3 by exploring the impact of neural network depth on cross-domain recommendation systems. The research uncovers a "Goldilocks effect," demonstrating that a four-layer architecture strikes the optimal balance between model complexity and performance, outperforming deeper architectures. The findings provide practical guidelines for designing deep learning models that are both computationally efficient and accurate.

Chapter 5: Conclusion and Future Work The final chapter synthesizes the insights gained from the research and highlights its contributions to the field. In addition to summarizing the findings, this chapter identifies future opportunities for extending the proposed frameworks. Potential directions include incorporating Large Language Model (LLM) and Natural Language Processing (NLP) techniques to improve the extraction of user preferences from unstructured data such as reviews and textual feedback. Furthermore, exploring real-time applications and multi-domain recommendations with implicit feedback data can further broaden the impact and applicability of these systems.

This thesis makes contributions to the development of accurate and efficient recommender systems, addressing challenges like the cold-start problem and providing new insights into model design and optimization.

1.2 Motivations

The importance of recommender systems in our daily lives is undeniable. From the moment we wake up and encounter recommendations for our morning routines to the time we choose a movie to watch before bed, these systems play a crucial role in shaping our decisions. Their ability to offer high-accuracy recommendations is vital across various domains. Beyond enhancing user experience, accurate and inclusive recommender systems have the potential to improve decision-making, reduce choice overload, and democratize access to information and services. These qualities make them indispensable across diverse fields such as education, healthcare, and e-commerce.

Conventionally, recommendation systems have relied on methods such as Collaborative Filtering (CF) and Content-based Filtering (CBF). While effective in certain contexts, these approaches face significant limitations in addressing modern challenges. For example, CF requires sufficient historical data about user-item interactions, making it ineffective in cold-start scenarios where such data is unavailable. Meanwhile, CBF relies heavily on manually crafted item features, which may not capture the full range of user preferences. It is also important to note that both methods are typically restricted to a single domain, which overlooks the potential for insights to be gained from cross-domain behaviors and interactions. The inability to address data sparsity and cold-start issues highlights the need for more advanced methods.

A promising direction lies in cross-domain recommendation systems, which can leverage interconnected user behaviors and preferences across seemingly unrelated domains—such as choosing a book to read, selecting a beverage, or deciding on clothing. For instance, the type of music you listen to or the books you read might influence your mood or preferences for activities and drinks. An advanced recommendation system that integrates these cross-domain inputs can capture these complex behavioral patterns, offering richer and more personalized suggestions. This underscores the need for research not only into domain-specific systems but also into cross-domain recommendation systems capable of bridging these interdependencies.

The growing popularity of AI-driven personalization, as seen in platforms like Netflix, Amazon, and Spotify, further highlights the demand for more sophisticated recommendation systems. However, a significant challenge in developing such systems is the cold-start problem, where limited or no information is available about certain users. This lack of data makes it difficult to deliver accurate recommendations. For example, without knowledge of a user's musical preferences, how can a system suggest suitable songs? Addressing this issue is particularly critical for cross-domain recommendation systems, which must combine data from multiple domains while respecting user privacy. Privacy concerns add another layer of complexity, as individuals may hesitate to share personal data across platforms or domains.

Despite ongoing efforts to address the cold-start problem, the potential of cross-domain recommendations in tackling this issue has been underexplored. This gap motivated my research to develop a system that can transfer knowledge from domains with abundant user information to those where information is scarce. Specifically, my work focuses on creating a cross-domain recommender system for tasks such as recommending books, music, movies, and items in the Kindle Store. Unlike traditional methods that rely heavily on domain-specific data, my research introduces a novel approach involving a transformation matrix to map relationships between domains. This matrix allows the system to transfer knowledge effectively, even for cold-start target domains. By employing a deep neural network with an attention mechanism to generate this transformation matrix, my approach offers a significant advancement in the field of cross-domain recommendation systems, paving the way for more inclusive and accurate personalized experiences.

A key inspiration for this research was the work by Zhu et al. on the Personalized Transfer of User Preferences for Cross-domain Recommendation [73], which introduced a meta-network-based framework to address the cold-start problem through personalized bridges between domains.

1.3 Problem Statement

Recommender systems deal with significant challenges in generating personalized recommendations for new users and items with a limited amount of interaction data. These limitations are particularly pronounced in cold-start scenarios, where the lack of historical data prevents accurate and relevant recommendations.

The term cold-start originates from mechanical systems, such as automobile engines, which struggle to perform optimally under cold conditions until they warm up. Similarly, a recommender system in a cold-start scenario struggles to generate accurate predictions due to the absence of sufficient interaction data. In both cases, the "cold-start" refers to suboptimal performance during an initial phase, where the system must operate under conditions of limited or incomplete data. The problem manifests itself due to the heavy reliance of modern recommendation algorithms on historical interaction data to infer user preferences or item relevance. It is impossible for these systems to identify patterns, make associations, or make meaningful recommendations without adequate data. The nature of the cold-start problem reflects the broader challenges of information sparsity and uncertainty, making it a critical bottleneck in personalized recommendation processes.

The implications of the cold-start problem are far-reaching. For instance, an e-commerce product may fail to reach its intended audience if the system lacks the interaction data necessary to identify relevant user groups, resulting in missed revenue opportunities. Streaming platforms struggle to recommend new content effectively to first-time users, often resorting to generic or trending suggestions that fail to capture individual preferences. Providing personalized treatment recommendations for new patients without historical data will likely result in suboptimal outcomes, potentially undermining trust in the healthcare system. The classified advertising platforms also face challenges when new ads are posted without sufficient user engagement, causing them to be overlooked despite their relevance, as older, data-rich ads dominate recommendations. In these examples, we can see how the cold-start problem affects not only system performance but also user satisfaction and trust across domains.

This research tackles the cold-start problem by introducing a personalized cross-domain transfer framework that leverages deep learning techniques to enhance recommendation accuracy. By transferring user preferences across domains, this approach aims to address the limitations of existing methods, ensuring adaptability and scalability in sparse data scenarios.

1.4 Research Contributions

This research makes important contributions to the field of recommender systems by addressing the persistent cold-start problem through innovative deep learning techniques. The key contributions of this work are as follows:

- **Tackling Cold-Start with Deep Personalized Transfer of User Preferences**: This research is the first to apply deep meta-learning and encoding to uncover user behavior patterns across domains. The proposed approach advances the state-of-the-art of model accuracy, as measured by five key evaluation metrics, and establishes a new benchmark for addressing the cold-start problem.
- **Layer Depth Matters**:
  - Empirical Insights into CDR Architectures: New insights and empirical evidence are provided regarding the design of optimal deep learning architectures for cross-domain recommender systems. This study emphasizes the importance of selecting the appropriate layer depth, demonstrating how simpler architectures can achieve superior performance and improve scalability in sparse data scenarios.
  - Extensive Experimental Validation: To verify the effectiveness of the proposed deep transfer learning framework, we conducted extensive experiments on diverse Amazon datasets, including movies, music, and books. In both cold-start

and warm-start scenarios, these experiments validate the framework's ability to generalize across domains and improve recommendation accuracy. Additionally, growing evidence supports the effectiveness of fine-tuning deep learning models. Fine-tuning enhances the performance of cross-domain recommender systems, reinforcing the importance of model adaptation.

1.5 Publications

The research presented in this thesis includes findings that have been submitted to peer-reviewed journals. The first paper has been accepted, while the second one is currently under review. The details are as follows:

Omidvar, S., & Tran, T. (2023). Tackling cold-start with deep personalized transfer of user preferences for cross-domain recommendation. **Published in** *International Journal of Data Science and Analytics (IJDSA)*, 10 pages, DOI: 10.1007/s41060-023-00467-9, Springer, November 2023.

Omidvar, S., & Tran, T. (2024). Layer Depth Matters: The Goldilocks Effect in Deep Cross-domain Recommendation. **Under Review** of *International Journal of Data Science and Analytics (IJDSA)*.

Chapter 2

Background Information, Literature Review, and Related Work

2.1 Introduction

This chapter serves as the foundation for understanding the methodologies and advancements in CDR that form the basis of this thesis. It begins with Background Information, providing an overview of the challenges inherent in recommender systems, such as data sparsity and the cold-start problem. The subsequent sections explore three critical technological pillars: Deep Neural Network (DNN), which enables the modeling of complex user-item relationships; the Attention Mechanism, which dynamically prioritizes important features for personalized recommendations; and TL, a paradigm that leverages knowledge from resource-rich domains to improve recommendation accuracy in underrepresented ones. A complete discussion on recommender systems follows, focusing on their evolution and their expanding role in mitigating information overload across diverse applications. This chapter also includes a Literature Review synthesizing related works and identifying research gaps. The chapter concludes by spotlighting three recent, impactful contributions to the field of CDR.

2.2 Background Information

This section establishes the theoretical foundation for understanding the key advancements and methodologies relevant to this thesis. As the field of CDR continues to evolve, leveraging state-of-the-art technologies such as DL, attention mechanisms, and TL have become essential to address longstanding challenges like data sparsity and the cold-start problem. These challenges hinder the performance of traditional recommender systems, particularly in domains where user-item interaction data is insufficient for generating reliable predictions [17]. By integrating advanced machine learning techniques, CDR has demonstrated the potential to transfer knowledge effectively across domains [25], thereby enhancing recommendation accuracy in underrepresented settings.

The section is organized into three subsections that delve into technologies underpinning CDR advancements. The first subsection explores DNN, which has revolutionized the field by modeling complex relationships between users and items through hierarchical feature representations. The second subsection introduces the Attention Mechanism, a powerful concept in deep learning that dynamically prioritizes relevant features to improve recommendation quality. The third subsection focuses on TL, a method that enables knowledge transfer from rich source domains to sparsely populated target domains, forming the backbone of many modern CDR systems. Lastly, the section concludes with an overview of Recommender Systems, tracing their evolution and highlighting their critical role in mitigating information overload across various domains. Together, these subsections provide a background for understanding the technologies and methodologies driving this research.

2.2.1 Deep Neural Network

The DNNs represent a class of Machine Learning (ML) models that reflect the structure and functionality of the human brain [65]. These networks are made up of layers of interconnected nodes that are organized into three layers: an input layer, a layer that is hidden,

and a layer that is output. An input neuron processes the data by applying a weighted sum to calculate the activation function and then sends this to a subsequent layer so that the network can learn how to represent the data in a more complex manner. It is the hierarchical nature that makes them so powerful since each layer extracts higher-level features from the raw input data progressively [42]. For example, the initial layers of image recognition may detect simple patterns such as edges, while deeper layers may identify more abstract structures such as shapes or objects. As a result of this layered architecture, they are capable of modeling complex nonlinear relationships, making them highly effective for tasks such as computer vision, natural language processing, and speech recognition.

Several aspects of DNNs have advanced over the years, cementing their position as a cornerstone of modern artificial intelligence. By introducing nonlinearity to the network with activation functions, such as Rectified Linear Unit (ReLU), sigmoid, and tanh, the network is able to model complex patterns [33]. With innovations such as Exponential Linear Unit (ELU) and Gaussian Error Linear Unit (GELU), issues like vanishing gradients have been addressed, leading to a better convergence of the model [7, 38]. Additionally, a number of optimization techniques, such as Stochastic Gradient Descent (SGD), Adam, and RMSprop, are designed to iteratively minimize a loss function, which can then be used to refine the parameters of the network [21]. The learning rate schedules and momentum features are also added as enhancements to further speed up and stabilize the training process [21]. In the following chapters, we will provide in-depth discussions of the specification of the DNN, detailing the selection of activation functions, optimization methods, and additional techniques used to enhance performance and achieve desired results.

Methods such as dropout, weight decay, and batch normalization have been developed to prevent overfitting and improve generalization [30]. This technique prevents the network from becoming overly dependent on specific features in the training data. The training process involves two key phases: forward propagation and backpropagation. Forward propagation involves traversing the input data through the network to make predictions. Following this, the backpropagation calculates the gradient of the loss function with respect to each parameter, adjusting weights to minimize prediction errors. To achieve satisfac-

tory results, the training requires a considerable amount of computational resources and large datasets. Tuning hyperparameters, such as choosing the optimal number of layers, neurons, and learning rates, remains a significant challenge and often requires a strong understanding of the domain [51,67]. In the following chapters, we will discuss regularization methods, training strategies, and hyperparameter choices used in this thesis to achieve robust performance.

2.2.2 Attention Mechanism

A significant advancement in machine learning is the Attention Mechanism, which allows models to focus on the most relevant parts of input data dynamically [48]. Since it originated in the field of NLP [59], it has become widely used across a variety of fields, including computer vision [18], speech recognition [41], and reinforcement learning [26]. The basic idea of the attention mechanism is to allocate different levels of importance, or weights, to input features, prioritizing critical information while downplaying irrelevant or redundant data [48]. Traditional machine learning models, on the other hand, treat all input features equally, thereby limiting the model's ability to identify patterns in complex datasets [6]. This limitation is addressed by attention mechanisms that weigh the input based on the most relevant features to the task at hand [6]. It is particularly useful in contexts where input sequences vary in length or importance, like sentences in text processing or frames in video analysis [66].

This thesis uses scaled dot-product attention, which compares input features with all other input features in the sequence, as a widely used implementation of attention. A query, a key, and a value are the three key components of this process. Similarity scores are computed by comparing queries with keys, then scaled and normalized to produce attention weights. The output is generated by applying these weights to the values. The model can thus focus on specific features while maintaining a global context of the entire input. A notable architecture leveraging the attention mechanism is the Transformer [59], even though this thesis does not focus on it. By combining self-attention with recurrent and

convolutional operations, the transformer architecture improves computation efficiency by processing inputs in parallel [59]. BERT and GPT are examples of models that are powered by this innovation, and have since been adapted for vision tasks and other applications [15]. Different attention mechanisms, including self-attention, multi-head attention, and cross-attention, have been developed to extend their applications [20]. Through self-attention, models are able to focus on different parts of the same input sequence, capturing intricate dependencies and relationships. Multi-head attention enhances this capability by allowing the model to attend to multiple aspects of the data simultaneously, improving its ability to represent complex patterns. Cross-attention facilitates interactions between different input sources, such as aligning images with text in multimodal tasks. This thesis focuses on self-attention, exploring its capabilities and applications in detail.

There are many benefits to attention mechanisms. The learned attention weights can provide insights into which parts of the input were influential in making predictions by allocating resources to the most relevant features. Additionally, they improve model performance when long-range dependencies or contextual understanding are required. Attention mechanisms, however, can be computationally expensive, especially when used for long input sequences, since their complexity grows quadratically with length [62]. This challenge has been addressed through innovations such as sparse attention and efficient transformers, which reduce computational overhead while preserving performance. This research leverages attention mechanisms for ratings, with details provided in the following chapters.

2.2.3 Transfer Learning

The concept of TL describes machine learning techniques that use knowledge gained from one task or domain to improve performance on a related, but distinct, task or domain [61]. In situations with limited data or resources, it allows models to draw insight from a source domain with abundant information, which is particularly helpful when the target task lacks sufficient data. As opposed to traditional machine learning methods, which train models

from scratch for each task, this kind of learning makes it possible for models to generalize across domains by reusing pre-trained features [19]. It assumes that certain patterns or structures learned in one domain are applicable in another. As an example, a model trained on large-scale image data that recognizes generic objects may be able to transfer its learned features, such as edges and textures, to an image classification model designed to recognize medical images. As the target model is initially equipped with pre-trained weights from the source model, it significantly reduces the amount of data used for training and improves convergence.

The core of TL is the optimization of a model that has been trained on the source domain and its adaptation to the target domain. During pre-training, the model parameters θ are optimized to minimize a loss function L_S on the source domain $\mathcal{D}_S = \{(x_S, y_S)\}$, where x_S are input features and y_S are the labels or classes. With this approach, early layers learn general features such as edges or syntax, while deeper layers learn domain-specific features. In the transfer phase, the model is fine-tuned on the target domain $\mathcal{D}_T = \{(x_T, y_T)\}$ by minimizing the target loss function L_T:

$$\theta_T = \arg\min_{\theta} \frac{1}{N_T} \sum_{i=1}^{N_T} L_T(f(x_{T_i}; \theta), y_{T_i}),$$

where N_T is the number of target samples, $f(x_T; \theta)$ is the model's prediction, and L_T is the target task's loss. In feature extraction, the earlier layers of the network are frozen so that their general-purpose features are leveraged, while the final layers are updated. Fine-tuning, on the other hand, adjusts all layers with a slower learning rate to retain previously learned knowledge while adapting to the target domain. When the source and target domains differ significantly, techniques like adversarial domain adaptation minimize distribution discrepancies [11] $\mathcal{D}(P_S, P_T)$, where P_S and P_T are the data distributions of the source and target domains, respectively.

In addition to accelerating training and improving performance in the target domain, the use of TL goes beyond simply reusing models. By enabling the discovery of latent relationships between seemingly unrelated domains, it is a useful tool for tackling com-

plex tasks with limited data. In cross-modal applications, the alignment of embeddings facilitates tasks like image captioning and visual question answering by connecting text and images. The ability to combine knowledge from diverse domains has also proven useful in multi-domain learning, allowing a single model to generalize across tasks with partial overlaps in features. In fields such as autonomous systems, personalized healthcare, and resource-constrained environments, these capabilities highlight the versatility and widespread potential of these technologies. Multi-domain learning has also benefited from its ability to combine knowledge from diverse domains, enabling a single model to generalize across tasks with partial overlaps in their feature spaces. Using multiple source domains to increase prediction accuracy in the target domain can further extend this concept. By integrating insights from diverse domains—such as books, music, and movies—a richer and more detailed understanding of user behaviors or data patterns can emerge, reducing biases inherent in single-source models. Although this approach introduces additional challenges, such as aligning domain distributions, managing redundant information, and increasing computational complexity due to the integration of multiple domains, these complexities are manageable and will be discussed in further detail in the following chapters.

2.2.4 Recommender Systems

We can consider recommender systems as a specialized branch of information retrieval systems, designed to predict user preferences and generate personalized suggestions. By directing users towards relevant content, products, or services, these systems address the growing problem of information overload in digital environments [53]. Users interact with and discover content through recommendation systems across a variety of domains, including e-commerce, entertainment, healthcare, and education. Recommender systems analyze user preferences and behaviors, often represented as a sparse matrix of user-item interactions, to generate predictions for items the user has yet to interact with.

A combination of techniques is used to achieve this, broadly categorized as CF, CBF,

and hybrid methods. In CF, it is assumed that users with similar historical preferences will behave similarly in the future. Based on shared interactions, this approach identifies patterns by analyzing either user-user or item-item similarities [56]. As an example, users who rate the same movies highly are grouped together, and their collective preferences are used to suggest additional movies. Similarly, items frequently co-rated by users are recommended to others who interact with related items. A significant advantage of it is its ability to leverage the collective wisdom of the user community and to uncover implicit relationships within large datasets. Conversely, CBF focuses on the attributes of items to make recommendations tailored to individual user preferences [28]. By analyzing metadata such as genres, keywords, or textual descriptions, it builds a profile for each user based on the characteristics of items they have previously engaged with. For instance, a user who enjoys action movies might receive suggestions for other action films, even if no other users share similar preferences. It achieves meaningful results by focusing on detailed metadata, though it relies heavily on user-item relationships to ensure personalization. Hybrid methods synthesize the strengths of both mentioned methods, employing various strategies to achieve more accurate and robust recommendations. Combining community-driven insights from CF with the personalized focus of CBF, hybrid approaches deliver more diverse and precise recommendations customized to individual needs while taking advantage of the broader structure of user-item interactions [4].

The advances in machine learning and data processing have significantly improved the performance of recommender systems. Matrix Factorization (MF) techniques, such as Singular Value Decomposition (SVD) are widely used to reduce the dimensionality of user-item matrices and identify latent factors explaining user preferences [45]. More recently, deep learning approaches have emerged as powerful tools for modeling complex user-item interactions. For capturing sequential behavior, contextual information, and long-range dependencies, methods such as autoencoders, recurrent neural networks, and transformers have been applied. Despite their effectiveness, they face data sparsity, scalability, fairness, bias, privacy, and cold-start challenges [27]. As a result of the fact that most users interact with a limited number of items, data sparsity arises, making it difficult to draw meaning-

ful conclusions from the data. Scalability happens in real-world scenarios where systems must handle millions of users and items. Fairness ensures that recommendations do not favor popular items or certain user groups disproportionately [9]. Biases can be caused by imbalanced training data or by algorithmic design, which can result in suboptimal recommendations. Privacy concerns are increasingly important as recommender systems rely on sensitive user data [22]. Lastly, the cold-start problem, which is the focus of this paper, refers to the inability to make accurate recommendations for new users or items due to a lack of historical information.

2.3 Literature Review

Despite significant advances, the existing memory-based and MF methods have limitations that could restrict their scope and performance, according to Feng et al. [12]. One of the limitations is the lack of information regarding user preferences and item characteristics relative to the large number of potential users and items within the system, known as the data sparsity problem.

The cold-start problem is a specific case of data sparsity that happens when the recommender system faces new users or items in the system. Many attempts have been made to address cold-start situations without using a rich source domain, like using exploration and exploitation techniques to identify interesting items for users by weighing items' popularity with entropy, as Silva et al. [57] suggested. In another approach, Li et al. [40] suggested an online CF method incorporating dynamic regularization and considering real-time information to employ the neighborhood factor and determine the changes over time. Regarding sparsity, Margaris et al. examined factors related to rating prediction accuracy in sparse CF datasets and indicated that recommending items that achieve higher prediction values than others can reduce recommendation accuracy and negatively affect a recommender system's performance in some cases [44]. Koohi and Kiani utilized a clustering-based CF approach to predict the unrated entries to overcome data sparsity using available rating

values from the user-item matrix [35]. Nevertheless, this approach does not provide diverse recommendations or leverage data across multiple domains to address preferences. It thus results in popularity bias, a phenomenon whereby popular items are recommended over new and less popular ones.

The notion of TL can be used in the context of CDR. It has been demonstrated that the concept of TL can be applied successfully to the recommendation task. A CF method based on CDR was proposed by Pan et al. [50] to alleviate data sparsity difficulties. However, it does not address transferring knowledge between heterogeneous settings, e.g., movies, music, and books. To address this problem, Moreno et al. [47] proposed a TL technique that extracts knowledge from multiple domains containing rich data (e.g., movies and music) and generates recommendations for a sparse target domain (e.g., games) by filling in missing values in the target domain rating matrix. Despite its ability to solve the data sparsity issue, overfitting is still challenging because it cannot detect embedded patterns in user behavior.

Another subfield of machine learning is meta-learning, also known as "learning to learn," which focuses on developing algorithms and techniques that enable machines to learn how to learn. Neural network meta-learning has a long history [60]. Its potential as a driver to advance the frontier of the contemporary deep learning industry has led to an explosion of recent research [23]. To this end, Yongchun Zhu et al. proposed a transfer-meta framework for CDR (TMCDR), which has both a transfer stage and a meta stage [72], but it trains a common bridge as existing bridge-based methods do.

2.4 Related Works

In this section, we examine three prominent papers on CDR - Embedding and Mapping for Cross-domain Recommendation (EMCDR) [43], Deep Framework for Cross-domain and Cross-system (DCDCSR) [70], and Personalized Transfer of User Preferences for Cross-domain Recommendation (PTUPCDR) [73] which serve as the stepping stones of this

thesis. These approaches employ unique methodologies to address the challenges of data sparsity and cold-start problems. DCDCSR combines MF and DNN to accurately map latent factors across domains, considering user and item sparsity levels to improve recommendation accuracy. With EMCDR, latent spaces are embedded and mapped using Multi-layer Perceptron (MLP), effectively connecting user and item representations. Lastly, PTUPCDR uses personalized bridge functions to transfer preferences between domains based on user characteristics generated by a meta-network. In this section, we explore their contributions, methodologies, and comparative impacts on addressing the challenges.

2.4.1 Embedding and Mapping framework for Cross-domain Recommendation

Man et al. [43] proposed the EMCDR framework to address the persistent problem of sparse data in recommender systems. With this approach, recommendation quality is improved by transferring knowledge from one domain to another, thus overcoming the limitations of sparse data within a single domain.

There are three interconnected stages in the framework. To begin with, latent factor modeling is employed to project users and items from both the source and target domains into their respective latent spaces, achieved using MF and Bayesian Personalized Ranking (BPR). MF captures user-item interactions by factoring the user-item interaction matrix into low-dimensional latent vectors that represent underlying preferences and characteristics. On the other hand, BPR emphasizes ranking-oriented optimization, constructing preference pairs from implicit feedback to prioritize recommendations based on their ranking.

Following latent factor modeling, the framework establishes a relationship between the latent spaces of the source and target domains through a mapping function. Using this function, latent representations across domains can be bridged, allowing knowledge to be transferred. Two mapping approaches are proposed within the framework: a linear

mapping function and a nonlinear mapping function based on a MLP. The linear mapping approach employs a transfer matrix to establish a direct linear relationship between the latent spaces, providing computational efficiency but limited flexibility. Conversely, the MLP approach captures complex, nonlinear relationships, offering greater adaptability to diverse data distributions. With backpropagation, the nonlinear mapping function is optimized for robust performance even when cross-domain dependencies are intricate.

The final stage of the framework involves generating recommendations in the target domain. By using mapped latent factors from the source domain, the framework predicts ratings or preferences for users and items in the target domain, despite limited interaction data. To compensate for the sparsity of data in the target domain, this mapping-based inference system uses knowledge from the source domain. Through a selective training strategy, the framework mitigates the effects of noise and overfitting caused by sparse data by leveraging only active users and popular items. This limitation is addressed in DCD-CSR. Finally, the framework's performance is rigorously evaluated on real-world datasets, including MovieLens-Netflix and Douban, with experimental conditions outlined in the following chapters.

Their framework introduces foundational ideas for addressing data sparsity through knowledge transfer, and it effectively bridges latent spaces between domains using either a linear or nonlinear mapping function; however, its reliance on static mapping limits its adaptability to user-specific nuances. On the other hand, our framework leverages a personalized bridge mechanism that dynamically and not statically adjusts to each user's unique interaction patterns, enhancing precision and robustness in recommendations.

2.4.2 A Deep Framework for Cross-Domain and Cross-System Recommendations

The DCDCSR proposed by Zhu et al [70] is another approach to address the persistent data sparsity problem by combining latent factor modeling with a sparsity-aware mapping

mechanism. It builds on the strengths of earlier methods while introducing innovations to improve the accuracy and adaptability of cross-domain and cross-system recommendations. The framework integrates MF with a fully connected DNN for mapping user and item latent factors across domains or systems. This approach is implemented in three interconnected phases.

In the first phase, MF decomposes the user-item interaction matrices from the source and target domains. Upon decomposition, user and item latent factor matrices are obtained for both domains. The framework supports multiple MF models, including Maximum-margin Matrix Factorization (MMMF), Probabilistic Matrix Factorization (PMF), and BPR. The MMMF algorithm optimizes the predictive trace margin, the PMF algorithm assumes Gaussian noise in observed ratings, and the BPR algorithm maintains consistency between observed and predicted ratings. The results of these models are used to generate latent factors that provide the basis for further analysis.

The second phase introduces benchmark factors to bridge the latent spaces of the source and target domains. Considering the sparsity of user and item ratings, these benchmark factors are calculated by integrating the latent factors from both domains. Latent factors for users and items common to both domains are weighted according to their sparsity degrees, thereby ensuring that more reliable factors, based on higher rating density, have a greater influence. To identify the most similar entities from the source, the framework uses cosine similarity, which combines the latent factors of the entities into benchmark factors that are appropriate for the target domain. As soon as the benchmark factors are established, both the target latent factors and benchmark factors are normalized to a common range in preparation for network training.

The network serves as the core mapping mechanism. As a fully interconnected architecture, it aims to capture complex, non-linear relationships between latent factors in both the source and target domains. Mapping loss is minimized through the use of feedforward and backpropagation processes. An activation function of tan-sigmoid is used in this network, and it is trained using gradient-based optimization. Once the training is completed,

the mapped latent factors are denormalized to restore their original scale.

In the final phase of the framework, the mapped user latent factors in the target domain are fixed after being processed by the network. Subsequently, the item latent factors in the target domain are further optimized using MF. Through this refinement, the item factors are aligned more closely with the updated user factors, improving the accuracy of the predicted ratings. As soon as the user and item latent factors are determined, the predicted ratings for user-item pairs in the target domain are calculated using the dot product of their respective vectors. Based on the user's preferences in the target domain, these predicted ratings are then used to rank the items for each user. As shown in Table 2.1, the comparison highlights the differences and similarities between the two approaches.

Their model incorporates sparsity-aware mechanisms to enhance latent factor mapping, making it more resilient to the challenges posed by sparse data. Although its benchmark factors and sparsity weighting are innovative for balancing cross-domain latent factors, this approach assumes uniform importance for all latent factors within a sparsity group. However, our model uses a mechanism that assigns dynamic importance weights to each interaction, ensuring that the most relevant factors are prioritized. Additionally, our personalized bridge functions further refine the mapping process, offering superior adaptability to individual user preferences.

2.4.3 Personalized Transfer of User Preferences for Cross-domain Recommendation

The PTUPCDR by Zhu et al. [73] presents an innovative approach to addressing the cold-start problem in recommender systems by transferring user preferences. Unlike conventional methods which employ the same preference bridge for all users, it generates personalized bridge functions tailored to each user, facilitating a more nuanced and effective transfer of preferences.

A meta-network is at the core of this model, which learns to produce personalized

Table 2.1: Comparison of DCDCSR and EMCDR models.

Aspect	DCDCSR	EMCDR	Similarities	Differences
Objective	Combines MF and DNN for CDR and CSR	Embedding and mapping framework for CDR using MF and MLP	Both aim to address data sparsity in CDR by transferring knowledge across domains.	DCDCSR employs benchmark factors and sparsity degrees, while EMCDR uses explicit latent space mappings.
Latent Factor Models	Utilizes MF-based models (MMMF, PMF, BPR)	Uses MF and BPR models	Both use MF and BPR for latent factor modeling to build embeddings.	DCDCSR incorporates additional sparsity-driven benchmark factors for mapping latent factors.
Mapping Mechanism	Employs DNN with sparsity-adjusted benchmark factors	Linear or MLP-based mapping	Both map latent spaces across domains to align embeddings.	DCDCSR uses a more detailed sparsity-driven benchmark adjustment; EMCDR's mapping is simpler but flexible.
Handling Sparsity	Adjusts for sparsity degrees in benchmark factor generation	Selects entities with sufficient data for robust mapping	Both account for sparsity but in different ways.	DCDCSR quantifies sparsity using specific metrics, whereas EMCDR avoids sparsity by preselecting dense entities.
Architecture	Multi-phase: MF Modeling, DNN Mapping, Recommendations	Steps: Latent Factor Modeling, Space Mapping, Recommendations	Both adopt multi-step frameworks to enhance recommendations.	DCDCSR is more complex, incorporating additional normalization and denormalization processes.
Task	Supports both CDR and CSR	Primarily focuses on CDR	Both frameworks can potentially handle similar recommendation scenarios.	EMCDR explicitly discusses user-shared and item-shared scenarios, while DCDCSR handles CSR more comprehensively.

bridge functions by applying embeddings derived from the source domain to individual users. The meta-network incorporates user traits that can be transferred and extracted from a characteristic encoder based on an attention mechanism. The encoder identifies relevant source domain interactions by assigning importance scores to items based on their contribution to cross-domain knowledge transfer. The framework captures the unique relationships between user preferences in both domains by compressing this information into a representation that reflects individual preferences.

The customized bridge function generated by the meta-network transforms the user's embedding from the source domain into a representation appropriate for the target domain. As a result of this transformation, the embedded representation serves as the initial representation for users in the target domain to solve cold-start scenarios. To stabilize the learning process of the meta-network, it employs a task-oriented optimization strategy, which directly optimizes the performance of the ultimate recommendation task. With this approach, the inaccuracies in user embeddings are mitigated, ensuring the robustness and generalizability of traditional mapping-based optimization methods.

The framework operates in three main stages. In the pre-training stage, latent spaces for each domain are learned independently. During the meta-training stage, the characteristic encoder and meta-network are trained to generate personalized bridge functions. In the initialization stage, the transformed user embeddings are applied as initial representations for cold-start users in the target domain. This methodology allows it to excel not only in extreme cold-start scenarios but also in warm-start scenarios where initial embeddings can be fine-tuned with subsequent interactions. The experimental evaluation conducted on real-world datasets such as the Amazon review dataset, demonstrates its superiority over baseline methods EMCDR and DCDCSR.

Their framework significantly relies on personalized bridge functions and attention-based characteristic encoders. However, its meta-network relies on a single-stage mapping process that may not fully exploit transferable characteristics when user interaction histories are highly diverse. In contrast, our framework differentiates itself by introducing

a deep meta-network capable of iterative refinement through task-specific optimizations. This approach enables the generation of more precise user embeddings tailored to cold-start and warm-start scenarios, ensuring improved recommendation quality across various user groups.

Chapter 3

Tackling Cold-Start with Deep Personalized Transfer of User Preferences for Cross-domain Recommendation

3.1 Introduction

This chapter focuses on the development and evaluation of our DPTUPCDR model, designed to tackle the cold-start problem in recommendation systems. It begins with a short review of the cold-start challenge and its significance in cross-domain settings, followed by a detailed description of the architecture, emphasizing its personalized bridge functions, deep characteristic encoder, and meta-network components. The chapter then outlines the experimental setup, including datasets, evaluation metrics, and baseline models, before presenting a thorough analysis of the results, highlighting the model's performance improvements over state-of-the-art approaches. Finally, the chapter concludes with a discussion of the findings, addressing the strengths, limitations, and potential avenues for

further enhancements.

As explored in previous chapters, recommender systems are an effective technology that alleviates the problem of overloading information provided to users [34]. Researchers began exploring CF techniques to make personalized recommendations in the 1990s [54]. Since then, recommendation systems have been actively studied, applied, and expanded in all fields of academia and industry in recent years [34]. Consequently, several other CF recommendation systems were developed over the next few years, including Amazon's "Customers who bought this also bought" feature in 1999 and Netflix's recommendation system in 2006, which offered a million-dollar prize for improving its accuracy [37].

It is common in real-world application scenarios that only a small number of users can rate and review items for a large number of items [52], a problem known as data sparsity, which reduces model recommendation accuracy [71]. To address this problem, CDR [39] has been introduced, which aims to transfer knowledge from an informative source domain to the target domain [73]. Not only is it capable of handling data sparsity, but it can also deal with new items and new users (the cold-start problem). The cold-start problem is a long-standing problem in recommender systems where systems cannot recommend relevant items to users due to the lack of adequate information [16]. As a comparison, the warm start problem occurs when a recommendation system has some information about a new user or item but not enough to give an accurate and personalized recommendation.

Most existing CDR methods assume all users share the same relationships between user preferences in the source and target domains [14,31,43]. For this reason, these methods will use a common bridge function between the source and target domains. To enhance this idea and address users' different tastes, Zhu et al. proposed using a personalized bridge for each user [73]. This idea states that there is simply a map function that allows users to move from the source domain to the target one. Towards this end, we seek to enhance the framework developed by Zhu et al. [73]. Based on the characteristic embeddings in the source domain they introduced, we generate personalized bridges for each user using a four-layer neural network. As shown in Figure 3.1, the deep meta-learner uses the user's characteristics

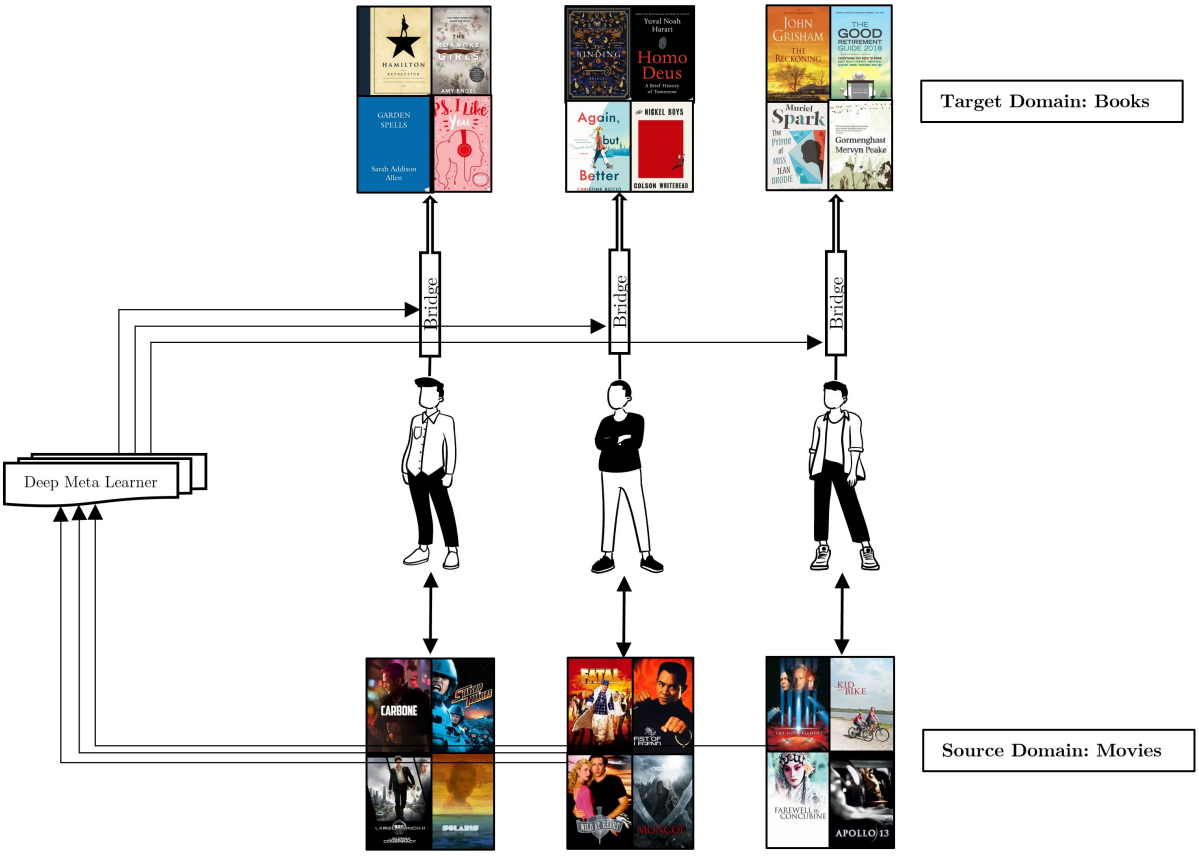

Figure 3.1: The extended version of PTUPCDR using a deep meta-network

from the source domain movies to generate personalized book recommendations. In the proposed DPTUPCDR model, since the meta-learner captures each user's taste pattern in the source domain and generates a personalized bridge for each user, it can be an effective method for cold-start users with no interaction history in the target domain. As a result of our observations, we demonstrate that our model is useful in real-world applications.

This research has yielded three primary contributions, which can be summarized as follows:

1. To the best of our knowledge, we are the first to apply deep meta-learning and encoding to reveal user behavior patterns across domains and significantly improve

the state-of-the-art model accuracy on two key metrics.

2. We conducted extensive experiments on different Amazon datasets for movies, music, and books to verify the effectiveness of the proposed framework.

3. Our study contributes to the growing body of evidence supporting the efficacy and potential of fine-tuning to improve deep learning performance in solving complex problems.

3.2 Model

In this section, we formulate the problem and present the framework. It should be noted that the majority of the information in this section and the terminology are the same as those in the EMCDR and PTUPCDR models, but there will be a discussion of the changes made to create a deep model. The primary distinction between our work and the existing models is the introduction of a deep learning-based architecture utilizing neural networks combined with the One Cycle Learning Rate (OneCycleLR) scheduling method, aiming to enhance training efficiency and model convergence.

Suppose $\mathcal{U} = \{u_1, u_2, \cdots\}$ and $\mathcal{V} = \{v_1, v_2, \cdots\}$ are the set of users and the set of items, respectively, and matrix $\mathcal{R}$ represent the interaction between these two sets. Accordingly, $r_{ij} \in \mathcal{R}$ indicates the rating given to item v_j by user u_i. Considering that we have source and target domains, we refer to $\mathcal{U}^s$, $\mathcal{V}^s$, and $\mathcal{R}^s$ as the set of users, the set of items, and the rating matrix for the source domain, respectively. Meanwhile, we represent the set of users, the set of items, and the rating matrix for the target domain by $\mathcal{U}^t$, $\mathcal{V}^t$, and $\mathcal{R}^t$, respectively. Although we assume for simplicity that the interaction between users and items is based on a rating system, further details will be discussed in the following sections. Because the source and target domains differ, $\mathcal{V}^s$ and $\mathcal{V}^t$ have nothing in common. On the other hand, some users participate in both domains, and we represent this intersection with $\mathcal{U}^o = \mathcal{U}^s \cap \mathcal{U}^t$.

After this brief terminology, our main objective is to use the source domain to improve prediction accuracy in the target domain to minimize the loss function, which is formulated as below:

$$\mathcal{L} = \min_{\mu} \frac{1}{|\mathcal{R}_o^t|} \sum_{r_{ij} \in \mathcal{R}_o^t} (r_{ij} - \hat{r}_{ij})^2 \tag{3.1}$$

where $\mathcal{R}_o^t = \{r_{ij} | u_i \in \mathcal{U}^o, v_j \in \mathcal{V}^t\}$ denotes the interactions of overlapping users in the target domain, $\hat{r}_{ij}$ is the predicted rating given by user u_i to item v_j, and μ is the set of parameters used. The loss function, which measures how well or poorly a model performs, is used with an optimization method to minimize prediction error. As the loss drops during training, the model becomes more proficient at identifying items with high ratings for each user, which signifies a strong affinity between the user and the item, paving the way for personalized recommendations to choose relevant items for each user in the target domain.

The latent factor model, a generalization of CBF, assumes that the factors responsible for the user's choice of an item need not be explicitly known [46]. As a result, users and items can be defined by their corresponding latent factors by transforming them into dense vectors. Suppose $u_i^d \in \mathbb{R}^k$ and $v_j^d \in \mathbb{R}^k$ are the embeddings of the user u_i^d and item v_j^d, respectively, where k denotes the dimensionality of embeddings and $d \in \{s, t\}$ represents the domain label. The list of users' sequential interactions in the source domain is $S_{u_i} = \{v_{t_1}^s, v_{t_2}^s, \cdots, v_{t_n}^s\}$ where n denotes the number of interacted items and $v_{t_p}^s$ indicates the interacted item in the source domain at timestamp t_p.

3.2.1 Deep Characteristic Encoder

As a first step in generating the personalized bridge function, it is necessary to capture the personalized transferable characteristics of users based on their interactions with items in the source domain since cold-start users have not interacted with items in the target domain. Although these characteristics contribute to knowledge transfer, not all are equally

important because some may introduce noise or fail to differentiate users from one another. In other words, to categorize users more effectively, we will seek characteristics that will cause more variance to differentiate users.

We aim to utilize an attention mechanism [64], a central concept in modern deep learning and natural language processing. As explained in previous chapters, a fundamental principle behind attention mechanisms is to give more weight to important characteristics than to less important ones. To this end, we extend its definition for recommendation systems and assign a weight to item embeddings by performing a weighted sum:

$$p_{u_i} = \sum_{v_j^s \in S_{u_i}} a_j . v_j^s \tag{3.2}$$

where:

- $p_{u_i} \in \mathbb{R}^k$: Transferable characteristic embedding of user u_i in the source domain.
- u_i: A specific user from the set of users for whom recommendations are generated.
- S_{u_i}: The set of items in the source domain that user u_i has interacted with or consumed.
- $v_j^s \in \mathbb{R}^k$: Embedding vector of item v_j in the source domain.
- a_j: Attention score (weight) assigned to item embedding v_j^s, representing its importance or relevance to user u_i.
- k: Dimensionality of embedding vectors in the latent space.

Accordingly, irrelevant items will have lower weights based on this mechanism. To calculate the attention weight a_j for item v_j, we exponentiate the output of $h(v_j; \theta)$ and normalize it by the sum of exponentiated outputs over all elements in the set S_{u_i}. By normalizing the attention weights, we ensure that the resulting probability distribution

indicates the degree of attention assigned to each element in S_{u_i} with respect to v_j. In other words, equation 3.3 encapsulates the core attention mechanism, which facilitates enhanced modeling of contextual dependencies and informative feature selection within network architectures:

$$a_j = \frac{exp(h(v_j;\theta))}{\sum_{v_l^s \in S_{u_i}} exp(h(v_j;\theta))} \tag{3.3}$$

where $h(\cdot)$ represents the attention mechanism as a four-layer feed-forward network, and θ represents the set of its parameters.

3.2.2 Deep Meta-Network

Since the process of preference transfer needs to be personalized, we propose a deep meta-network that takes the user's transferable characteristics as input and generates a personalized bridge function between the user's embeddings in the source and target domains. With this bridge function, we can determine each user's preference in the target domain, especially for cold-start users. According to equation 3.4, the personalized bridge function w_{u_i} for user u_i is obtained by applying a function $g(\cdot)$ to the user's transferable characteristics p_{u_i} using a set of learned parameters ϕ:

$$w_{u_i} = g(p_{u_i};\phi) \tag{3.4}$$

The proposed deep meta-network $g(\cdot)$, a four-layer feed-forward neural network, takes two arguments p_{u_i} and ϕ. The first argument represents the transferable characteristic embedding of user u_i in the source domain, calculated in equation 3.2. The second argument represents a set of parameters associated with the function $g(\cdot)$ learned during the training process and defines how the transformation from p_{u_i} to w_{u_i} is performed.

When we have both the embedding of user u_i^s and its personalized bridge function w_{u_i}, we can use a simple linear function $f(\cdot)$ to multiply w_{u_i} by u_i^s in order to transform it into

$\hat{u}_i^t$:

$$\hat{u}_i^t = f_{u_i}(u_i^s; w_{u_i}) \tag{3.5}$$

The parameters used in equation 3.5 are clearly defined as follows:

- $\hat{u}_i^t \in \mathbb{R}^k$: The transformed embedding of user u_i in the target domain, obtained by mapping the source domain embedding through the personalized bridge function.
- $f_{u_i}(\cdot)$: A user-specific linear transformation function parameterized by w_{u_i} to project the user's embedding from the source domain into the target domain.
- $u_i^s \in \mathbb{R}^k$: The embedding of user u_i in the source domain, capturing the user's preferences or characteristics based on source-domain interactions.
- w_{u_i}: The personalized bridge function parameters specific to user u_i, defining how their embedding in the source domain is transformed into the embedding in the target domain.

To clearly illustrate the relationship between equations 3.2 and 3.5, it should be noted that the embedding u_i^s in equation 3.5 corresponds exactly to the personalized transferable characteristic embedding p_{u_i} calculated by equation 3.2. Specifically, we first compute the user's embedding in the source domain, p_{u_i}, by aggregating item embeddings weighted by attention scores as shown in equation 3.2. Next, this embedding is transformed via the personalized bridge function w_{u_i}, generated by the deep meta-network described in equation 3.4. Finally, by applying the linear transformation $f_{u_i}(\cdot)$ in equation 3.5, the user's source domain embedding is projected into the target domain embedding space, yielding the transformed embedding $\hat{u}_i^t$.

The linear function $f_{u_i}(\cdot)$ in equation 3.5 is defined as EMCDR and PTUPCDR models, we specifically implement $f(\cdot)$ as a linear transformation. To fit the dimensions required

for this linear transformation, we reshape the parameter vector $w_{u_i} \in \mathbb{R}^{k^2}$ into a matrix $w_{u_i} \in \mathbb{R}^{k \times k}$. Thus, the practical implementation of the linear function is:

$$f_{u_i}(u_i^s; w_{u_i}) = w_{u_i} \cdot u_i^s \tag{3.6}$$

Here, $u_i^s \in \mathbb{R}^k$ represents the user's embedding in the source domain, and w_{u_i} denotes the personalized parameters reshaped as a $k \times k$ projection matrix. Consequently, the transformed embedding $\hat{u}_i^t$ is obtained as the result of this personalized linear mapping.

A possible solution would be to minimize the distance between the transformed embedding of a user in the target domain and the actual embedding of that user. Based on existing techniques [31], this optimization is formulated as follows:

$$\mathcal{L}' = \sum_{u_i \in \mathcal{U}^o} \left\| \hat{u}_i^t - u_i^t \right\|^2 \tag{3.7}$$

As shown in equation 3.7, an optimization algorithm can be used to minimize loss $\mathcal{L}'$ by iteratively updating each user's personalized bridge function. However, Zhu et al. [73] suggested that since some users only have limited interactions, the user's embedding u_i^t may not be reasonable and accurate enough. The best approach is to optimize $\mathcal{L}$ in equation 3.1 instead of $\mathcal{L}'$ in equation 3.7, where $\mu = \{\theta, \phi\}$ and $\hat{r}_{ij}$ will be calculated as follows:

$$\hat{r}_{ij} = \hat{u}_i^t . v_j \tag{3.8}$$

Equation 3.8 calculates the interaction between the transformed user representation $\hat{u}_i^t$ and the item representation v_j using a simple dot product operation to quantify how well the user's transformed preferences align with the item's characteristics. If the result is negative or excessively large, we apply clipping to keep the score within a reasonable range.

Table 3.1: Statistics of the cross-domain tasks (Overlap denotes the number of overlapping users).

Task	Domain		Item		User			Rating	
	Source	Target	Source	Target	Overlap	Source	Target	Source	Target
Task 1	Movie	Music	50,052	64,443	18,031	123,960	75,258	1,697,533	1,097,592
Task 2	Book	Movie	367,982	50,052	37,388	603,668	123,960	8,898,041	1,697,533
Task 3	Book	Music	367,982	64,443	16,738	603,668	75,258	8,898,041	1,097,592

3.3 Experimental Evaluation

In this section, we present the experiments conducted to evaluate the performance of our proposed method. We first describe the datasets used, evaluation metrics, and experimental settings. Next, we compare our method's performance with the state-of-the-art approach, both before and after fine-tuning the hyperparameters. Additionally, we analyze the impact of hyperparameters on model performance. Our experimental approach will address the following research questions:

1. What is the significance of incorporating an auxiliary domain, and how does our model perform compared to the latest models?
2. How do hyperparameters impact results?

3.3.1 Dataset

Our study uses the Amazon-5cores dataset[1], a real-world public dataset containing user-item ratings for various products in different categories, including games, appliances, books, music, and movies. This dataset ensures data validity by including at least five ratings per

[1] https://cseweb.ucsd.edu/ jmcauley/datasets/amazon_v2/

user and item. This research focuses on movies, music, and books. The details and statistics of these cross-domain tasks are presented in Table 3.1, adapted from Zhu et al. [73], which summarizes the number of users, items, and ratings in each source and target domain. Overall, using a publicly available and diverse dataset such as Amazon-5cores allows us to test our proposed method's performance and generalization on a wide range of product categories and rating scenarios. This enhances the robustness and applicability of our findings to real-world recommendation systems. All experimental results presented in this and the next chapter, both for baseline models and our proposed model, are reported as the average over ten independent runs to ensure statistical reliability and reproducibility.

3.3.2 Evaluation Metrics

To evaluate the performance of our proposed method for the recommendation task, we used several commonly employed metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Precision, Recall, and F1 Score.

The MAE measures the average absolute difference between predicted and actual ratings, while the RMSE measures the root mean squared difference between predicted and actual ratings. These metrics are computed as follows:

$$MAE = \frac{1}{|\mathcal{T}|} \sum_{(i,j)\in\mathcal{T}} |r_{ij} - \hat{r}_{u,i}| \tag{3.9}$$

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(i,j)\in\mathcal{T}} (r_{ij} - \hat{r}_{u,i})^2} \tag{3.10}$$

where $\mathcal{T}$ represents the set of user-item pairs in the test set. These metrics are often used as optimization objectives during the training of recommendation models to directly improve the accuracy of predictions.

In addition to MAE and RMSE, we evaluated classification-oriented metrics: Precision, Recall, and F1 Score. These metrics assess the model's ability to correctly predict

user preferences when predicted ratings are rounded and clipped to a valid range. The definitions and calculations for these metrics are as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.11}$$

Precision measures the proportion of correctly predicted relevant items (*True Positives*) out of all items the model predicted as relevant (*True Positives + False Positives*). A higher Precision indicates that the model is making fewer false positive errors, meaning most of the items it recommends are genuinely preferred by users.

- **True Positives (TP):** The number of relevant items correctly identified as relevant by the model.
- **False Positives (FP):** The number of items incorrectly identified as relevant by the model, but they are not preferred by the user.

$$Recall = \frac{TP}{TP + FN} \tag{3.12}$$

Recall, also known as sensitivity, measures the proportion of all relevant items that the model successfully identified (*True Positives*) out of the total number of relevant items (*True Positives + False Negatives*). A higher Recall indicates that the model is capturing more of the truly relevant items, reducing false negative errors.

- **False Negatives (FN):** The number of relevant items that the model failed to identify as relevant.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.13}$$

The **F1 Score** is the harmonic mean of Precision and Recall, providing a single balanced measure that considers both false positives and false negatives. A higher F1 Score indicates that the model strikes a good balance between Precision and Recall, avoiding overemphasis on one at the expense of the other.

To address class imbalance in the data, we compute these metrics using a weighted average across all classes. Predictions are rounded to the nearest integer and clipped to a valid range of 1 to 5, ensuring consistency with the rating scale used in the dataset.

3.3.3 Baselines

We selected three methods for comparison with our proposed model. TGT is a simple MF model that uses only target-domain-specific features for training. As discussed in the previous chapter, EMCDR [43] adopts MF to learn embeddings and uses a common bridge function between all user embeddings from the auxiliary domain to the target domain. Finally, the state-of-the-art model PTUPCDR is an extension of EMCDR that uses personalized bridge functions to transfer knowledge between domains and learn both shared and domain-specific representations. The performance of our proposed method will be evaluated against these baseline methods using the evaluation metrics discussed earlier. The experimental results and analysis are presented in the following subsections.

3.3.4 Implementation Details

To ensure a fair comparison with previous methods and demonstrate the superiority of our proposed model, we used the same hyperparameters for all common ones as Zhu et al. [73]. The dropout probabilities of the deep encoder and deep meta-network were set to 10% and 20%, respectively. We used SGD [36] as the optimizer for training our model, along with the OneCycleLR scheduling policy [58]. OneCycleLR aims to achieve faster convergence, higher accuracy, and improved generalization of deep neural networks. To

enhance our model's robustness and prevent the "dying ReLU" problem, we utilized the ELU activation function in all layers.

3.3.5 Experimental Design

In this section, we will discuss the reasoning behind our experimental setup and justify our choice of parameters. By utilizing this setup, one can evaluate the model's performance accurately, gain valuable insights into its behavior, compare the results to the results of baseline models, and make meaningful conclusions.

The choice of test set size, denoted as β, is a critical factor in cross-domain recommendation experiments, representing the percentage of users in the test set that will be used for evaluation. In developing a recommendation model, the decision regarding β impacts the trade-off between having a large test set for evaluation and having a larger dataset for training. We systematically varied β between 20% and 50% of the total overlapping users in each domain to analyze its effect on the success of our framework. According to Table 3.2, our experiments suggest that the recommendation accuracy decreases as β increases, as larger test sets result in fewer training data, potentially limiting the model's ability to capture complex user preferences. A smaller β allowed the model to gain access to more training data, allowing it to generalize to the target domain more easily. On the other hand, there is a risk of overfitting, especially when the user-item interactions are limited. Even so, our results indicated that the model exhibited robustness across different β values, suggesting that it does not suffer from overfitting, thus justifying the choice of hyperparameters in the model.

3.3.6 Evaluation Results & Comparison

This study evaluated the proposed model, and compared it to three other models, on three different cross-domain tasks with different proportions of test users β as 20% and 50% of the total overlapping users, respectively. Evaluation metrics used in the study are the

Table 3.2: Average results of 3 cross-domain tasks with different β over different runs, with the best results in boldface. "Improve 1" denotes the relative improvement of PTUPCDR over EMCDR, while "Improve 2" indicates the relative improvement of DPTUPCDR over PTUPCDR. Similarly, "Improve 3" refers to the relative improvement of DPTUPCDR-FT (DPTUPCDR after fine-tuning) over PTUPCDR.

Tasks	β	**Metric**	**TGT**	**EMCDR**	**PTUPCDR**	**Improve 1**	**DPTUPCDR**	**Improve 2**	**DPTUPCDR-FT**	**Improve 3**
Task 1	20 %	MAE	4.4803	1.2350	1.1504	6.86%	1.1170	2.90%	**0.8926**	22.41%
		RMSE	5.1580	1.5515	1.5195	2.06%	1.4846	2.30%	**1.1781**	22.47%
	50%	MAE	4.4989	1.3277	1.2804	3.57%	1.2388	3.25%	**0.9728**	24.03%
		RMSE	5.1736	1.6644	1.6380	1.59%	1.6782	-2.45%	**1.3077**	20.17%
Task 2	20%	MAE	4.1831	1.1162	0.9970	10.68%	1.0242	-2.73%	**0.7974**	20.02%
		RMSE	4.7536	1.4120	1.3317	5.69%	1.3104	1.60%	**1.0623**	20.16%
	50%	MAE	4.2288	1.1832	1.0894	7.93%	1.0710	1.69%	**0.8313**	23.69%
		RMSE	4.7920	1.4981	1.4395	3.91%	1.3907	3.39%	**1.1093**	22.94%
Task 3	20%	MAE	4.4873	1.3524	1.2286	9.15%	1.1891	3.22%	**0.9162**	25.43%
		RMSE	5.1672	1.6737	1.6085	3.90%	1.5469	3.83%	**1.2143**	24.51%
	50%	MAE	4.5073	1.4723	1.3764	6.51%	1.3480	2.06%	**1.0357**	24.75%
		RMSE	5.1727	1.8000	1.7447	3.07%	1.8077	-3.61%	**1.4067**	19.38%

RMSE and MAE, as discussed in section 3.3.2. The results are presented in Table 3.2, along with bar charts in Figures 3.2, 3.3, 3.4, and 3.5 that display the RMSE and MAE of all models (except TGT due to its poor results) with β values of 20% and 50%, respectively. The figures show that our model shows superior performance by achieving lower RMSE and MAE scores than all other models, regardless of β values, illustrating its robustness and stability. The key difference between fine-tuning and no fine-tuning in Table 3.2 lies in the optimization of model hyperparameters. In the fine-tuning setting, we performed a grid search to select the best values for several hyperparameters, including learning rate, number of epochs, activation function, dropout rate, number of layers, and momentum.

Table 3.2 shows that the TGT model performs poorly compared to other models, and we can understand that using an auxiliary domain could significantly enhance the results.

Table 3.3: Average results of 3 cross-domain tasks with different β values, including Precision, Recall, and F1-Score.

Tasks	β	**Metric**	**EMCDR**	**PTUPCDR**	**DPTUPCDR**
Task 1	20	Precision	0.5293	0.5001	0.5547
		Recall	0.1806	0.3949	0.5023
		F1-Score	0.1852	0.4274	0.5069
	50	Precision	0.5339	0.5067	0.5410
		Recall	0.1699	0.4266	0.5180
		F1-Score	0.1697	0.4489	0.4974
Task 2	20	Precision	0.4656	0.4582	0.5410
		Recall	0.3088	0.3694	0.4310
		F1-Score	0.3313	0.3879	0.4353
	50	Precision	0.4629	0.4506	0.5194
		Recall	0.2777	0.3772	0.4234
		F1-Score	0.2924	0.3941	0.4293
Task 3	20	Precision	0.5260	0.5038	0.5496
		Recall	0.1502	0.3875	0.4879
		F1-Score	0.1511	0.4262	0.5017
	50	Precision	0.5395	0.5020	0.5297
		Recall	0.0864	0.3402	0.4281
		F1-Score	0.0625	0.3881	0.4583

Based on the evaluation results, the proposed model outperformed all other models, including the state-of-the-art PTUPCDR, by an average of 10% in terms of RMSE and MAE. The bar chart also demonstrates that DPTUPCDR is superior to all other models, as it consistently shows the lowest RMSE and MAE across all β values. Overall, the evaluation results indicate that the proposed model is more efficient than existing models in the field.

Additionally, Precision, Recall, and F1 Score were included in our evaluation, as shown in Table 3.3. These metrics allowed us to more fully understand the model's capability to identify relevant items efficiently, which is a reflection of its practical utility in real-world recommender systems. The proposed model consistently outperformed all other models across these metrics. The model is shown to be capable of both predicting user preferences

accurately as well as balancing precision and recall, which is critical to minimizing false positives and false negatives.

These additional metrics are being evaluated as they directly correlate with user experience in recommender systems. The precision of the recommendations reduces the risk of overwhelming users with irrelevant recommendations. Recall, on the other hand, ensures that a significant proportion of relevant items is identified, thus improving the coverage of the system. With the F1 Score, we can assess the performance of a model holistically by taking into account both aspects.

It should be noted that these results are less than those typically observed in simpler ML tasks, such as binary classification, which is because the problem at hand is far more complex, involving multiple classes derived from the rating scale, making it significantly more challenging for the model to predict the correct class. Furthermore, the evaluation is conducted using a cold-start scenario in which no prior user data exists in the domain of interest, affecting learning signal availability. Despite these difficulties, the model's adaptability and robustness shine through and underscore its ability to overcome domain discrepancies and deliver precise, relevant recommendations in diverse cross-domain recommendation tasks.

3.3.7 Impact of the Hyperparameters

The choice of hyperparameters significantly influences deep learning models' performance. Several hyperparameters need to be adjusted, and we conducted a series of experiments to examine the impact of different hyperparameter settings on the performance of the proposed model. As explained earlier, we used grid search to systematically explore different values, and the detailed findings for each hyperparameter are reported below.

Learning Rate. We used the OneCycleLR scheduling method to adjust the learning rate during training, which improved deep learning models' convergence speed and accuracy. We found that setting the maximum learning rate to 0.02 and the minimum learning

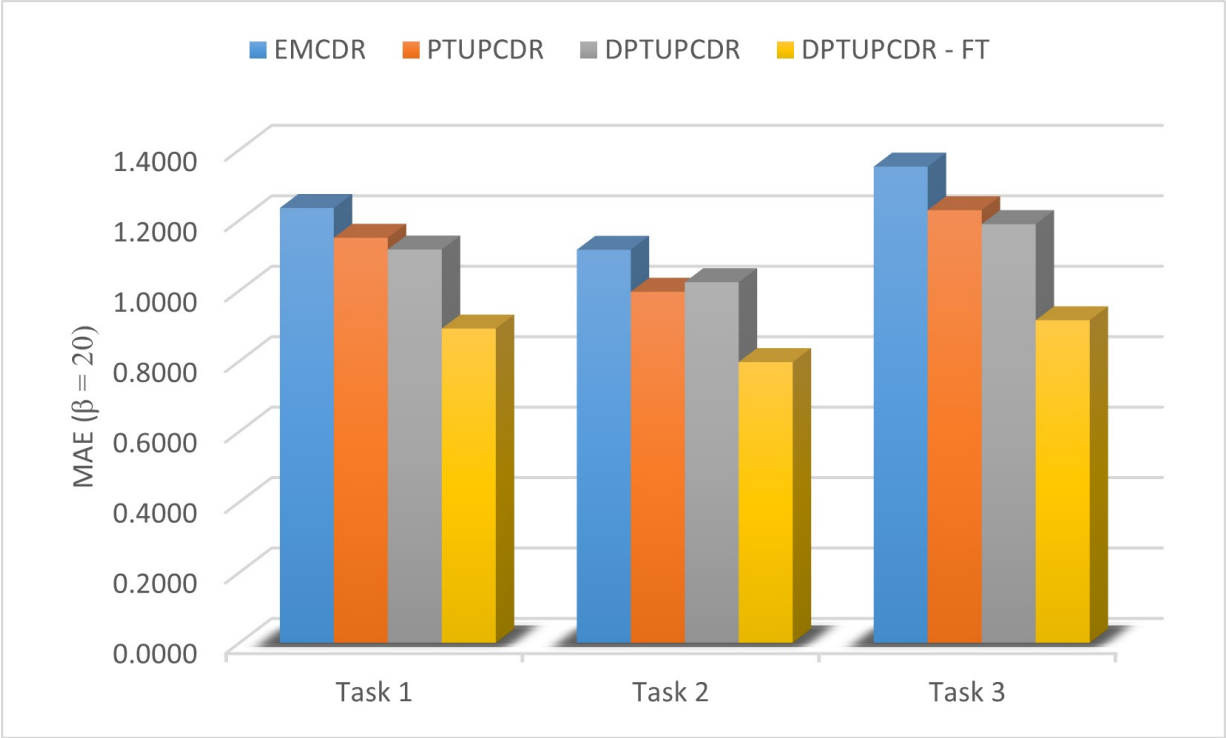

Figure 3.2: Comparative analysis of MAE across different models for three tasks at $\beta = 20$. Results indicate the improved performance of DPTUPCDR-FT in minimizing MAE across all tasks.

rate to 0.0001 achieved the best performance for our model. As we increased and decreased the starting learning rate, we found that the model could not converge within 15 epochs.

Epoch. By adding more epochs, we realized that the results did not improve but rather that the computation time increased. It is recommended that the OneCycleLR scheduling method be used with a cycle length of 15 epochs for all three tasks.

Activation Function. An activation function should be selected based on the specific problem to be addressed and the architecture of our DNN. Our experiments suggest that the ELU as the activation function improves accuracy compared to the ReLU. Its smoothness and ability to produce negative output values to adjust weights properly in deep models

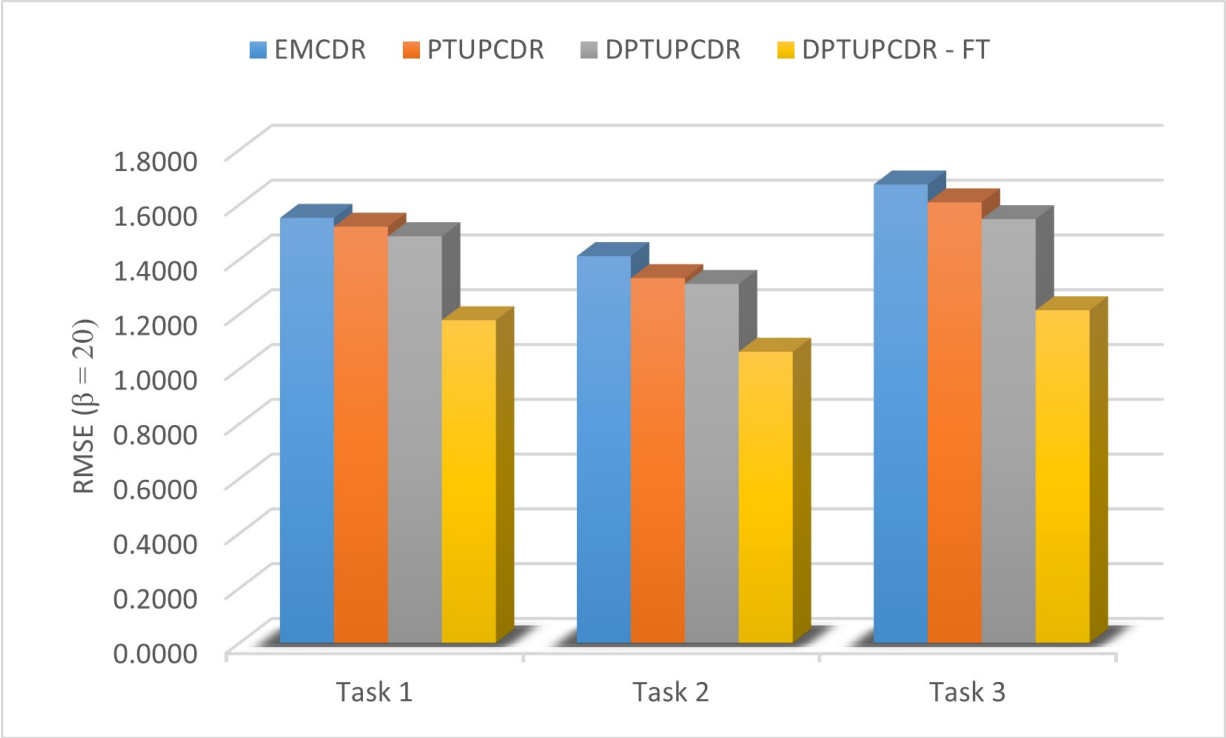

Figure 3.3: Comparative analysis of RMSE across different models for three tasks at $\beta = 20$. DPTUPCDR-FT demonstrates the lowest RMSE values, confirming its effectiveness.

explain its superiority over ReLU.

Dropout. After evaluating the model's performance for different β, we observed that increasing the dropout rate decreased performance. Therefore, we selected a dropout rate of 0.1 for the encoder and 0.2 for the meta-network.

Number of Layers. A neural network with many layers can learn complex patterns and relationships from data. The downside of having too many layers is that they can also result in over-fitting, where the model cannot generalize well to new data. Through empirical experimentation and testing, we have found that a neural network with four layers performs well.

Momentum. The momentum parameter, which is like a "memory" in a neural net-

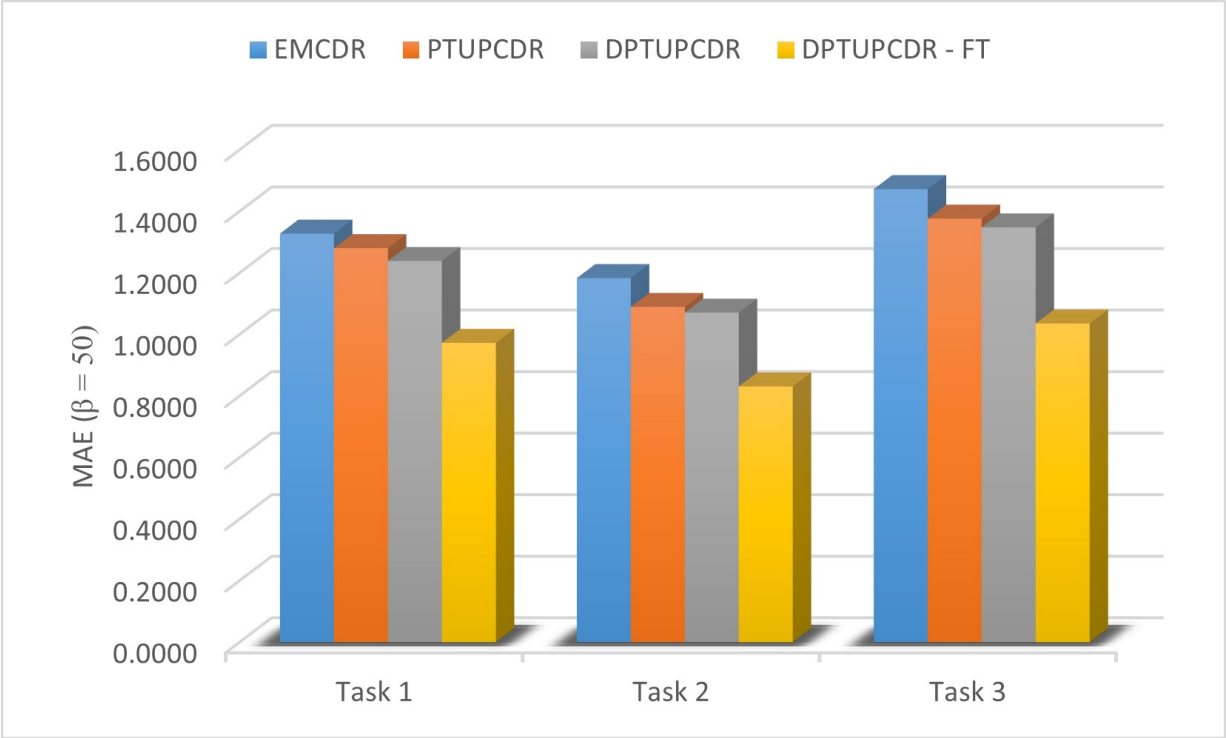

Figure 3.4: Comparative analysis of MAE across different models for three tasks at $\beta = 50$. DPTUPCDR-FT outperforms other models by consistently achieving lower MAE values.

work, can adjust the pace of the optimization process. Although momentum can speed up the optimization process, it may also lead to overshooting the optimal solution. With a momentum of 0.9, we achieved the best performance for our model.

These experiments illustrate the importance of hyperparameter tuning for deep learning-based CDR models. We find that careful selection and fine-tuning of hyperparameters is critical to achieving the best performance of models based on deep learning.

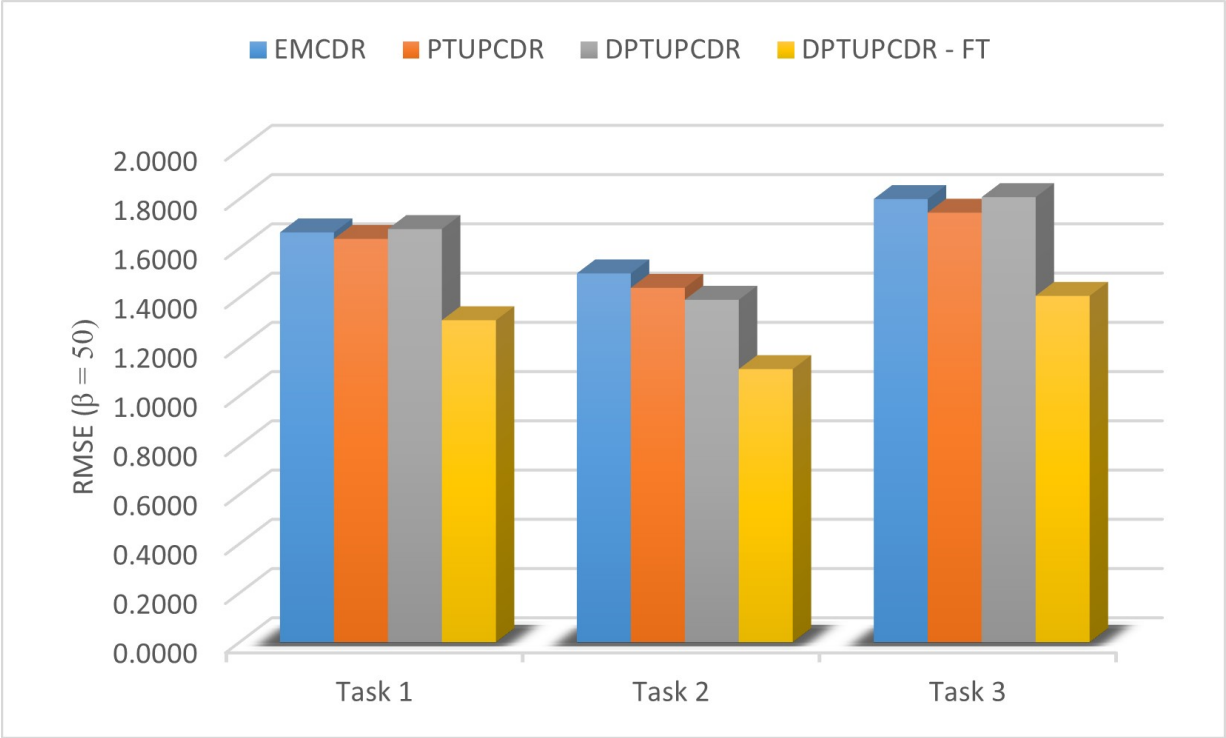

Figure 3.5: Comparative analysis of RMSE across different models for three tasks at $\beta = 50$. The results highlight the consistent reduction in RMSE achieved by DPTUPCDR-FT.

3.4 Discussion

Our study aimed to improve cold-start recommendations accuracy by utilizing personalized bridge functions to capture users' unique preferences. While the literature shows that most previous studies [12, 14, 31, 43, 69] have demonstrated the effectiveness of their methods in extreme cold-start scenarios using simple base models, these models are limited in their ability to extract users' characteristics since they rely on a single common bridge function and lack the sophistication needed to identify specific behaviors. To address this limitation, our model, as well as the one proposed by Zhu et al., utilizes a personalized bridge function to accurately capture users' preferences and provide more precise recommendations than previous models. While a general function that applies to all users can

increase the model's generalization, our study suggests that its simplicity can reduce the effectiveness of recommendations in the target domain. Therefore, using a personalized approach, as demonstrated by our model and Zhu et al.'s model, can provide more accurate recommendations by capturing users' unique preferences and behaviors.

In recommender systems, MAE and RMSE are widely regarded as the most important metrics because they directly quantify the difference between predicted and actual ratings, offering precise insights into a model's accuracy. While Precision, Recall, and F1 Score are more commonly used for classification tasks, we applied a threshold to define classes in our study to evaluate these metrics. This approach allowed us to demonstrate the superior performance of our model not only in terms of MAE and RMSE but also across these classification-inspired metrics. However, it is crucial to note that MAE and RMSE remain the most relevant metrics for recommendation systems, as they directly reflect the accuracy of predicted ratings, which is the primary goal of such systems. Including additional metrics provides a more comprehensive evaluation, but their importance should be considered secondary.

Additionally, our results highlight both the strengths and limitations of our approach regarding its applicability across different domains. Our method performs exceptionally well in entertainment-related areas such as movies, music, and books, where user preferences tend to be more consistent. In these cases, recommendation decisions largely rely on underlying user characteristics. However, our model may not generalize effectively to domains that involve high-stakes decisions, such as healthcare, finance, or critical decision-making tasks. In these contexts, recommendations must consider explicit domain knowledge, comply with regulations, and adhere to ethical standards.

Our proposed model and Zhu et al.'s model are the only ones currently using personalized bridge functions. However, our model differs from the state-of-the-art model in two key ways. Firstly, we utilized deep learning models to better and more efficiently understand users' source domain preferences and extract them for predicting their target domain preferences. This is because shallow models can only identify superficial patterns and cannot

identify latent characteristics, which deep learning models better capture. Secondly, we have carefully selected the OneCycleLR learning rate scheduler and ELU as the activation function in contrast to the state-of-the-art model, which uses ReLU and a fixed learning rate. Our study highlights the importance of carefully selecting the learning schedule and activation function to develop successful deep-learning models that achieve optimal accuracy and faster convergence. Furthermore, we found that Dropout and Momentum helped our model learn patterns rather than memorize them, improving accuracy. In contrast, the state-of-the-art model did not apply these principles. Our study provides insights into developing successful deep-learning models for cold-start recommendation systems that can improve accuracy and offer users more personalized recommendations.

3.5 Summary

This research investigates the concept of CDR, which transfers user preferences from a rich domain to a target domain using a personalized bridge function for each user. To learn the implicit underlying user's characteristic patterns and to address cold-start users with no interactions, we proposed the DPTUPCDR model. Our model outperformed existing state-of-the-art approaches in terms of MAE, RMSE, Precision, Recall, and F1-Score showcasing its effectiveness in addressing the cold-start issue. In addition, we conducted comprehensive experiments on Amazon real-world datasets and analyzed the impact of various hyperparameters on model performance. The results highlighted the significance of hyperparameter tuning and the advantages of using deep learning techniques, such as OneCycleLR learning rate scheduling and ELU activation function, to improve recommendation accuracy. Our research contributes to the field of recommender systems by addressing the cold-start problem, improving recommendation accuracy, and demonstrating the significance of hyperparameter tuning and deep learning techniques in developing successful CDR models.

Chapter 4

Layer Depth Matters: The Goldilocks Effect in Deep Cross-domain Recommendation

4.1 Introduction

This chapter extends the analysis of the DPTUPCDR model by focusing on the impact of architectural configurations, particularly layer depth, on its performance. Building upon the work presented in the previous chapter, this chapter explores additional experiments to validate the efficiency and scalability of the proposed model. A step further is the inclusion of all combinations of Music, Books, Movies, and Kindle Store domains in pairwise and vice-versa scenarios. The evaluation includes varying layer depths, ranging from 4 to 9 layers, and their effects on key metrics such as MAE and RMSE. Statistical analyses, including paired t-tests and Wilcoxon signed-rank tests, are conducted to assess the significance of the performance differences. Additionally, this chapter investigates computational costs, highlighting trade-offs between model complexity, accuracy, and training efficiency. The findings offer practical insights into optimizing the architectural design of deep learning-

based recommendation systems for real-world applications.

While the experiments aim to identify optimal architectural configurations, understanding theoretical guidelines for designing neural networks is equally important. Existing heuristics and theoretical approaches provide valuable insights into constructing neural networks but often fail to address specific challenges posed by deep architectures. According to Blum [1], a rule of thumb is that the hidden layer should have a size between the size of the input layer and the size of the output layer, while Boger et al. suggest setting a limit on the number of hidden nodes to capture 70-90% of the variance of the input dataset [2]. As helpful as these blind rules are as a starting point for constructing a neural network, they ignore several factors, such as the size of the dataset in the Blum strategy and the number of features and relations between them in the Boger strategy.

To address these limitations, researchers suggest using data-driven approaches to choose the optimal neural network architecture based on the characteristics of the dataset, such as dimensionality, distribution, and complexity. Among these approaches, Evolutionary Algorithm (EA), inspired by the natural selection process, iteratively evolves a population of potential architectures over multiple generations to discover configurations that exhibit superior performance on a given task and have been gaining traction as a powerful tool for determining the optimal number of nodes and layers [5, 13]. Using Genetic Algorithm (GA), Fiszelew et al. [13] calculated the fitness functions for all chromosomes in which each has a distinct structure and tried to minimize prediction error. Chhachhiya et al. [5] recommended using Particle Swarm Optimization (PSO) [10] in which each structure is a particle, and all particles seek the global optimum using social and cognitive components. Despite the fact that these EAs can provide appropriate results, calculating the fitness function for each is computationally expensive. Moreover, these algorithms often require many iterations to converge, heavily rely on defining a suitable fitness function, and can get stuck in local optima.

Alongside EA, algebraic methods such as SVD have also emerged as an influential avenue. As a linear algebra technique, it can be used to uncover hidden patterns in large

datasets and facilitate feature extraction and dimensionality reduction [55], which can inform the design of neural network architectures. A notable application is demonstrated by Cia et al. [3], who proposed using it to address time complexity and use dataset characteristics to estimate the number of hidden nodes in a single-hidden-layer feed-forward neural network. Although their method elegantly finds the number of hidden nodes in a single-hidden-layer neural network, it does not apply to a deep architecture, leaving the challenge of overfitting unsolved.

In our research, we have made two main contributions. First, we provide insights and empirical evidence regarding the optimal layer depth. Second, we contribute to understanding deep learning model design in the context of recommendation systems by emphasizing the advantages of simpler architectures.

4.2 Model

The proposed DNN architecture is designed with three main components: input, hidden, and output layers, each contributing to the overall transformation of raw data into meaningful predictions. Figure 4.1 illustrates the entire architecture, showcasing how input features flow through hidden layers and are processed to generate final outputs. The input layer consists of nodes $\{x_0, x_1, x_2, \ldots, x_n\}$, where each node corresponds to a feature of the input dataset. These features include numerical values, categorical encodings, or other pre-processed representations. This layer serves only as a conduit, forwarding the raw features to the hidden layers without any transformation.

The hidden layers, denoted by $\{a_{i,j}\}$, are the core computational units where most learning occurs. Each hidden layer applies a weighted sum of its inputs, combined with biases, and processes the result through a non-linear activation function, as depicted in Figure 4.2. These layers are fully connected, meaning each node in a layer is connected to every node in the preceding layer, ensuring comprehensive interaction between all features and learned parameters. The number of hidden layers and the number of nodes per layer

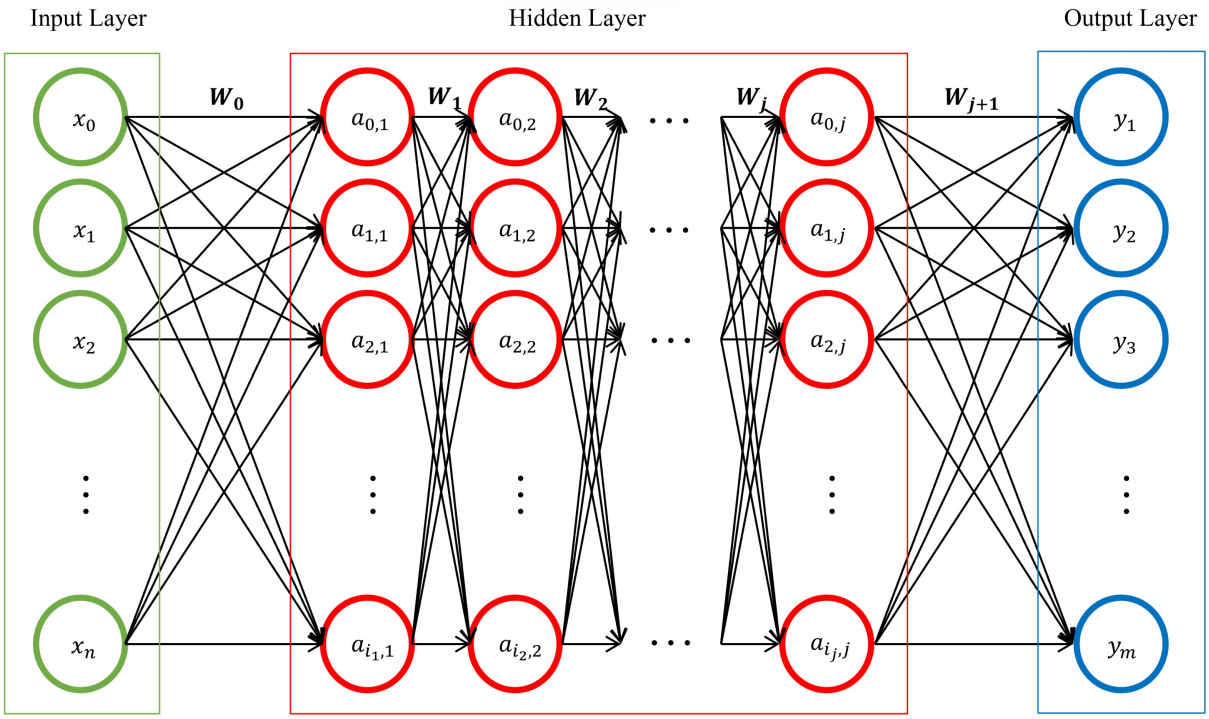

Figure 4.1: A general architecture of a deep feed-forward neural network with j hidden layers, W_k as weights for the k-th layer, $(a_{0,k}, a_{1,k}, a_{2,k}, ..., a_{i_k,k})$ as neurons of the k-th hidden layer, $(x_0, x_1, x_2, ..., x_n)$ as input, and $(y_1, y_2, ..., y_m)$ as output.

were optimized empirically to achieve a balance between model complexity and performance. Dropout regularization was applied during training to deactivate a random subset of neurons, reducing overfitting and improving generalization. In this thesis, when we refer to an *n-layer model*, we mean a network composed of $n-1$ hidden layers and one output layer, excluding the input layer. For example, a "4-layer model" consists of 3 hidden layers followed by 1 output layer.

The computations within each neuron follow a structured process. Inputs from the previous layer, represented as a vector $[x_1, x_2, \ldots, x_n]$, are aggregated into a single value using a weighted sum $W^{(i-1)T}X^{(i-1)}$ combined with a bias term. This weighted sum is

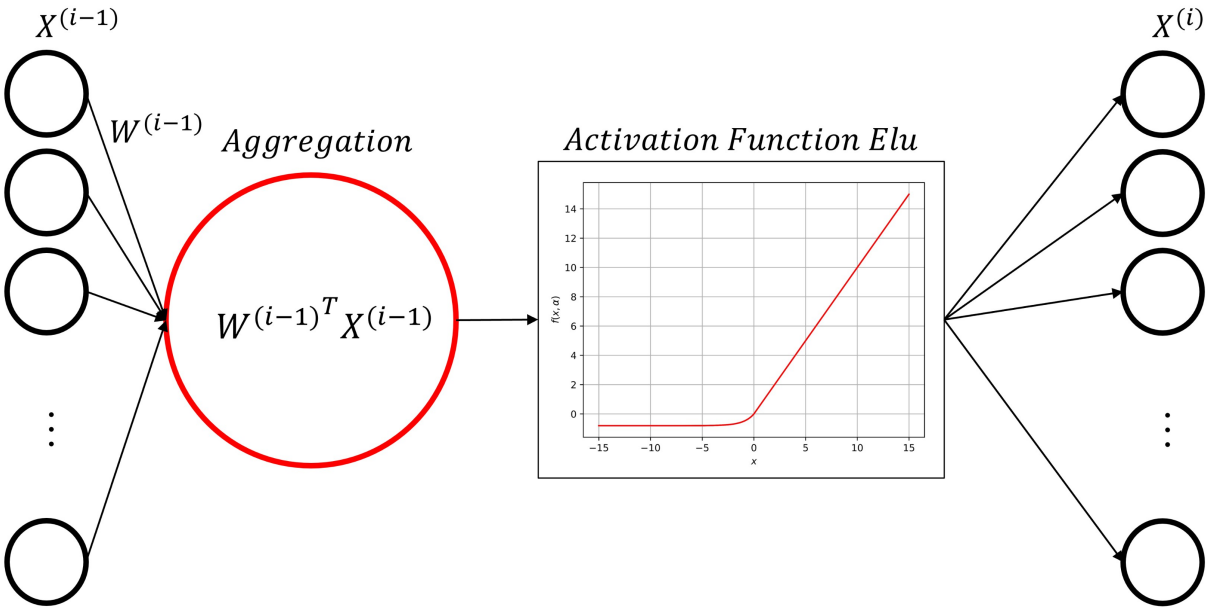

Figure 4.2: An overview of the stages involved in computing each neuron, including the aggregation of weights $W^{(i-1)}$ and input $X^{(i-1)}$, and the application of the ELU activation function to obtain the output $X^{(i)}$.

then passed through the ELU activation function:

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha(\exp(z) - 1) & \text{if } z \leq 0 \end{cases}$$

where α controls the saturation level for negative inputs. We choose this activation function for its ability to maintain effective gradient flow during backpropagation, addressing issues of vanishing gradients that can impede learning in deeper networks. Unlike simpler activation functions such as ReLU, it provides smooth transitions for negative values, enhancing convergence and stability during training.

The output layer consists of nodes $\{y_1, y_2, \ldots, y_m\}$, which correspond to the model's

predictions. Depending on the task, these outputs may represent probabilities for classification tasks, continuous values for regression, or other tailored outputs. The final architecture ensures that the model can capture complex relationships between input features while maintaining computational efficiency.

Figures 4.1 and 4.2 provide a detailed representation of the model's architecture and internal computations. Figure 4.1 emphasizes the hierarchical nature of the DNN, where data flows sequentially through layers, each learning progressively abstract representations. Figure 4.2 focuses on the neuron-level computations, demonstrating the weighted aggregation of inputs, the application of the ELU activation function, and the generation of outputs for subsequent layers. Together, these figures illustrate the transformation pipeline from raw features to final predictions, highlighting the interplay between architectural components and computational processes.

The architecture was optimized through systematic experimentation. Hyperparameters such as the learning rate, number of hidden layers, and nodes per layer were tuned to maximize performance on benchmark datasets. Dropout regularization was applied to prevent overfitting, with a dropout rate selected based on empirical evaluation. The training was conducted using backpropagation and SGD, with convergence monitored through validation loss and accuracy metrics.

The proposed architecture integrates domain-specific optimizations to address challenges in recommendation systems, such as sparse data and the cold-start problem. The model's design ensures scalability, adaptability, and robustness across different datasets and applications. By leveraging the strengths of ELU activation, dropout regularization, and fully connected layers, the architecture achieves a balance between complexity and performance, making it well-suited for practical deployment in recommendation systems.

As deep learning in recommender systems has grown in popularity [68], both industry and academia have competed to apply deep learning to a wide range of applications to solve many complex problems and provide state-of-the-art solutions [8]. By combining deep neural networks and using knowledge from a rich domain, we can make better recommen-

dations even in this cold-start situation, as previous studies have suggested [31, 43, 49, 73]. However, choosing an optimal architecture for a specific task is not trivial, as various models exist to consider [32]. As a result, it is imperative to spend the necessary time and effort selecting an appropriate architecture for our deep model. In this study, we aim to identify the optimal layer depth for achieving practical CDR by examining the performance of deep learning architectures with different layers.

4.3 Experiments

In this section, we perform a complete empirical investigation to verify the subtleties of our proposed model. Our exploration is structured to evaluate the architecture and effectiveness of the model in various application settings focusing on e-commerce environments. We start by presenting two research questions, which form the foundation of our experimental inquiry:

1. Does a four-layer architecture, identified as optimal in previous studies, maintain its scalability, efficiency, and performance across other application contexts, thus confirming our prior findings?

2. What advantages or limitations does the four-layer architectural choice present in e-commerce environments, and what are the implications of these findings in user experience enhancement and system efficiency in cold-start scenarios?

To begin with, we describe the datasets used in our experiments to lay the groundwork for our empirical analysis. Next, we explain the experimental settings, implementation details, and procedures employed to ensure thorough testing and validation of our model. Then, we explore the outcomes of our experiments and conduct a thorough statistical analysis to provide a reliable quantitative evaluation of the results and frame conclusions to understand the broader implications of our findings. This section ends with an analysis

of the complexity of our model, which offers valuable insights into its practical applicability and performance efficiency in real-world situations.

4.3.1 Dataset

In this section, we build upon the dataset described in the previous chapter, extending its use further to evaluate the proposed model's efficiency and scalability. It focuses on the categories of Books, Movies, Music, and the Kindle Store. We selected these domains because of their high levels of engagement and the diverse types of visual, auditory, and textual content they provide, which are essential for e-commerce recommendation systems. This dataset, which consists of user ratings provided on a scale of 1 to 5 for each item, was selected to ensure the utmost data richness and relevance, including products and users with at least five reviews.

Table 4.1 includes explanations and insights regarding the dataset's principal characteristics and attributes. In this table, the "Overlap" column, which indicates that users have interacted with items from the source and target domains, varies across tasks; the number of overlaps ranges from as few as 760 users in Tasks 6 and 11 to as many as 37,388 users in Tasks 2 and 7, suggesting a great user base that might lead to better predictions in the target domain.

The Books category in this dataset contains over 8 million reviews, while the Kindle Store category has around 1 million reviews. Literature trends and preferences can be examined through these categories. We can compare how readers behave and make choices in different media by comparing physical and digital formats. A comparison like this is important in the digital age because it enables us to understand the shift from traditional media to digital media. The analysis of these categories reveals patterns in reading habits, genre popularity, and the impact of digitalization on book consumption. Movies and TV in our dataset include approximately 1.6 million reviews, providing a rich source of information about Amazon users' entertainment preferences. This category is crucial for studying consumer trends in visual media and gaining insights into the popularity of films

Table 4.1: Statistics of the cross-domain tasks (Overlap denotes the number of overlapping users).

Task	Domain		Item	User		Rating
	Source	Target	Target	Overlap	Target	Target
Task 1		Music	64,443	18,031	75,258	1,097,592
Task 2	Movie	Book	367,982	37,388	603,668	8,898,041
Task 3		Kindle	61,934	2,754	68,223	982,619
Task 4		Movie	50,052	18,031	123,960	1,697,533
Task 5	Music	Book	367,982	16,738	603,668	8,898,041
Task 6		Kindle	61,934	760	68,223	982,619
Task 7		Movie	50,052	37,388	123,960	1,697,533
Task 8	Book	Music	64,443	16,738	75,258	1,097,592
Task 9		Kindle	61,934	68,084	68,223	982,619
Task 10		Movie	50,052	2,754	123,960	1,697,533
Task 11	Kindle	Music	64,443	760	75,258	1,097,592
Task 12		Book	367,982	68,084	603,668	8,898,041

and TV genres. Lastly, with over 1 million reviews, the Music category allows us to explore musical preferences and trends.

4.3.2 Experimental Settings

We begin by decompressing and parsing gzip-encoded JSON review files to construct a mid-level dataset comprising user IDs, item IDs, and ratings. Next, we match users across domains and item identifiers, creating a dataset, which is then divided into training and testing subsets using a predefined ratio. However, only the interactions with ratings exceeding a threshold of 3 are serialized into CSV format as the final processed dataset to

ensure that users' tastes are considered.

The core model implementation is built upon a two-part deep neural network architecture using PyTorch. The initial segment focuses on embedding functionalities, where unique identifiers for users and items are transformed into distinct embedded representations generated separately and concatenated. The latter segment of the model is a neural network divided into two principal sections: one for processing the embedded features using nonlinear ELU activation functions with dropout regularization and another for the reconstruction of inputs. This reconstruction process involves gradually upscaling the dimensionality of inputs to generate the final transformation matrix.

Optimization of the model is achieved through a gradient descent approach, emphasizing learning rate, weight decay, and momentum. A dynamic learning rate scheduler is implemented in tandem, adjusting the rate throughout the training process to optimize performance. Experimentation is conducted by varying the number of layers in both processing and reconstruction sections, ranging from four to ten layers, to evaluate the impact of network depth on the model's effectiveness and efficiency.

4.3.3 Evaluation Results

We conduct a comprehensive experimental analysis to evaluate the effects of different layer depths on transferring user preferences across domains. By examining the relationship between layer depths and model performance, we attempt to address the challenge of selecting an appropriate architecture. We evaluate the model using different proportions of test users β as 20% and 50% of the total overlapping users before and after fine-tuning for three independent tasks. Table 4.1 presents the statistical details of these tasks, while Table 4.2 presents the results using MAE and RMSE as evaluation metrics. We have also provided a visual representation of the results in Figures 4.4, 4.5, 4.6, and 4.7 via line charts for further understanding showing that the model's performance degrade when an excessive number of layers are added. Moreover, we can observe similar ascending and descending trends for two evaluation metrics, MAE and RMSE. It is important to note

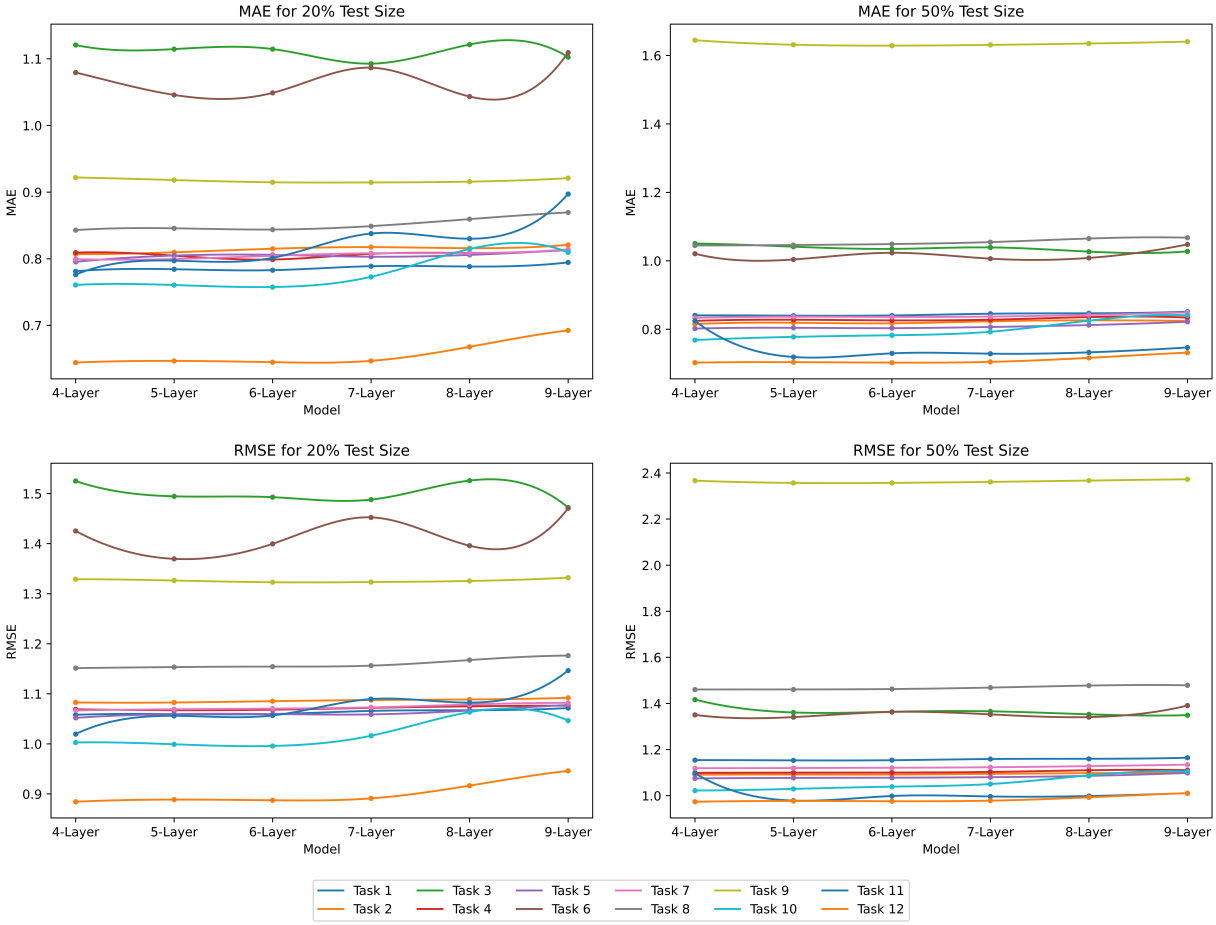

Figure 4.3: Comparative analysis of MAE and RMSE across 4- to 9-layer model configurations for 20% and 50% test sizes, across 12 different tasks.

Table 4.2: Comparison of our model with different numbers of layers ranging from 4 to 9 layers for all three cross-domain tasks with different β carried out for multiple runs after fine-tuning. Bold blue and red indicate the best and worst results, respectively.

Task	β	Metric	4-Layer	5-Layer	6-Layer	7-Layer	8-Layer	9-Layer
Task 1	20	MAE	**0.7813**	0.7842	0.7828	0.7888	0.7883	**0.7945**
		RMSE	**1.0580**	1.0596	1.0596	1.0658	1.0671	**1.0716**
	50	MAE	0.8404	**0.8396**	0.8402	0.8453	0.8466	**0.8511**
		RMSE	1.1542	**1.1530**	1.1537	1.1589	1.1599	**1.1643**
Task 2	20	MAE	**0.8069**	0.8096	0.8150	0.8174	0.8158	**0.8209**
		RMSE	1.0829	**1.0826**	1.0852	1.0875	1.0886	**1.0920**
	50	MAE	**0.8151**	0.8189	0.8173	0.8231	**0.8260**	0.8243
		RMSE	**1.0908**	1.0915	1.0915	1.0944	1.0976	**1.1002**
Task 3	20	MAE	1.1207	1.1146	1.1146	**1.0928**	**1.1214**	1.1025
		RMSE	1.5250	1.4945	1.4927	1.4879	**1.5259**	**1.4722**
	50	MAE	**1.0504**	1.0411	1.0347	1.0390	**1.0268**	1.0275
		RMSE	**1.4169**	1.3608	1.3640	1.3658	1.3528	**1.3491**
Task 4	20	MAE	0.8092	0.8046	**0.7988**	0.8074	0.8087	**0.8140**
		RMSE	1.0690	**1.0672**	1.0682	1.0719	1.0750	**1.0764**
	50	MAE	**0.8244**	0.8277	0.8258	0.8279	**0.8354**	0.8341
		RMSE	**1.0990**	1.0999	1.1003	1.1027	1.1101	**1.1103**
Task 5	20	MAE	**0.7953**	0.8048	0.8059	0.8030	0.8057	**0.8129**
		RMSE	**1.0520**	1.0595	1.0595	1.0588	1.0665	**1.0777**
	50	MAE	**0.8019**	0.8041	0.8031	0.8065	0.8121	**0.8217**
		RMSE	**1.0745**	1.0770	1.0777	1.0800	1.0858	**1.0985**
Task 6	20	MAE	1.0795	1.0458	1.0489	1.0867	**1.0434**	**1.1093**
		RMSE	1.4253	**1.3696**	1.3996	1.4524	1.3958	**1.4705**
	50	MAE	1.0207	**1.0038**	1.0234	1.0062	1.0083	**1.0479**
		RMSE	1.3504	**1.3407**	1.3632	1.3525	1.3407	**1.3908**
Task 7	20	MAE	**0.7994**	0.7996	0.8044	0.8082	0.8085	**0.8141**
		RMSE	**1.0666**	1.0692	1.0704	1.0729	1.0786	**1.0818**
	50	MAE	**0.8332**	0.8363	0.8358	0.8370	0.8417	**0.8489**
		RMSE	**1.1191**	1.1198	1.1210	1.1231	1.1280	**1.1342**
Task 8	20	MAE	**0.8429**	0.8457	0.8438	0.8489	0.8594	**0.8695**
		RMSE	**1.1513**	1.1534	1.1544	1.1562	1.1674	**1.1764**
	50	MAE	**1.0447**	1.0461	1.0489	1.0546	1.0650	**1.0674**
		RMSE	**1.4604**	1.4607	1.4625	1.4687	1.4776	**1.4787**
Task 9	20	MAE	**0.9219**	0.9181	0.9147	**0.9145**	0.9157	0.9209
		RMSE	1.3288	1.3263	**1.3228**	1.3232	1.3254	**1.3320**
	50	MAE	**1.6447**	1.6314	**1.6287**	1.6309	1.6349	1.6404
		RMSE	2.3667	**2.3568**	2.3569	2.3612	2.3670	**2.3725**
Task 10	20	MAE	0.7607	0.7606	**0.7576**	0.7729	**0.8145**	0.8096
		RMSE	1.0027	0.9992	**0.9958**	1.0162	**1.0634**	1.0465
	50	MAE	**0.7683**	0.7776	0.7823	0.7926	0.8251	**0.8406**
		RMSE	**1.0222**	1.0293	1.0390	1.0503	1.0880	**1.1047**
Task 11	20	MAE	**0.7761**	0.7971	0.8013	0.8378	0.8300	**0.8971**
		RMSE	**1.0195**	1.0560	1.0567	1.0894	1.0824	**1.1464**
	50	MAE	**0.8245**	**0.7189**	0.7294	0.7285	0.7323	0.7465
		RMSE	**1.0974**	**0.9785**	0.9984	0.9964	0.9983	1.0105
Task 12	20	MAE	**0.6443**	0.6467	0.6448	0.6468	0.6678	**0.6925**
		RMSE	**0.8843**	0.8887	0.8873	0.8911	0.9165	**0.9461**
	50	MAE	**0.7023**	0.7038	0.7023	0.7045	0.7163	**0.7317**
		RMSE	**0.9734**	0.9768	0.9757	0.9782	0.9926	**1.0110**

Table 4.3: The result of the paired t-test to determine the significant difference between the deep learning model with 4 and 5 layers before fine-tuning (BFT) and after fine-tuning (AFT)

T-Test	**BFT**	**AFT**
Pearson Correlation	0.99998226	0.999944181
df	11	11
t Stat	0.989121335	-2.128402479
$P(T \leq t)$ one-tail	0.17193113	0.028368355
t Critical one-tail	1.795884819	1.795884819
$P(T \leq t)$ two-tail	0.343862259	0.056736709
t Critical two-tail	2.20098516	2.20098516

that Figures 4.4, 4.5, 4.6, and 4.7 present the average results across all tasks and both β settings, providing a generalized view of the model's behavior.

Our experimental results indicate that the performance of the models degrades when the number of layers exceeds a limit of four or five, depending on the specific task, and even the use of an auxiliary rich domain does not mitigate the problem of overfitting. Based on the evaluation results presented in the table and line charts, it is evident that the proposed model with four or five layers outperforms the model with a higher number of layers, as it consistently shows the lowest MAE and RMSE across all beta values. As we proceed with this paper, we provide insights into the choice between a model with four or five layers and address the trade-off between model complexity and performance.

4.3.4 Computational Analysis

Even though deep learning has gained popularity, predicting the accurate time required to train a deep learning network remains a challenge [29]. On the other hand, it is possible to calculate the approximate computational cost for the feed-forward step by considering

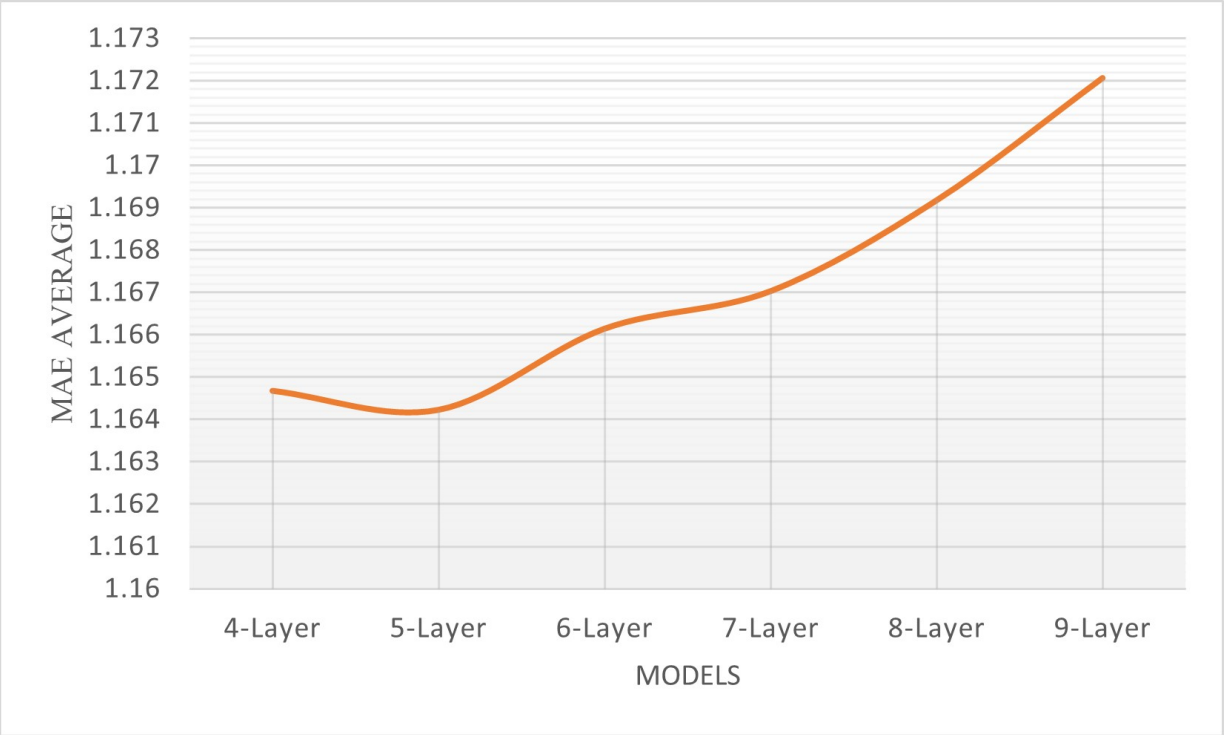

Figure 4.4: MAE across model depths (4 to 9 layers) before fine-tuning, showing performance decline with deeper models.

the total calculation time required for each neuron and multiplying it by the number of network runs. Given that different architectures have varying numbers of parameters, we should consider the time needed to update the weights in the back-propagation phase. Considering that all parameters may not fit entirely in the memory available, the system may need to perform memory transfers between different memory types, affecting the time required for training. Therefore, any estimation of training time in this context should be considered a rough estimate influenced by the hardware specifications. Accordingly, our model with four layers needs to learn 138 parameters, while a 5-layer model learns 182 parameters, and the number increases to 384 parameters for a nine-layer model. The number of parameters in each model is determined by summing the total weights and biases across all layers in the network. For each fully connected layer, the number of weights is

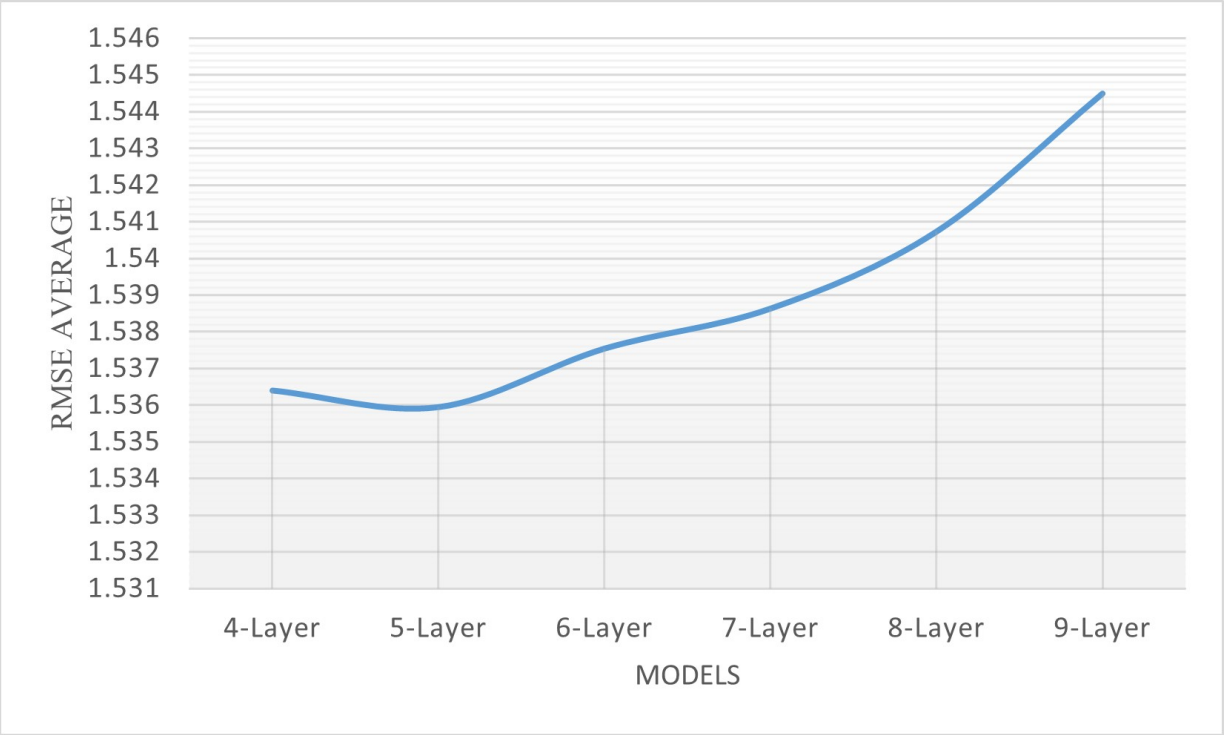

Figure 4.5: RMSE across model depths (4 to 9 layers) before fine-tuning, highlighting optimal performance at shallow depths.

calculated by multiplying the number of input neurons by the number of output neurons, and the number of biases equals the number of output neurons. That is, for a layer with n_{in} input neurons and n_{out} output neurons, the total number of parameters is given by $n_{\text{in}} \times n_{\text{out}} + n_{\text{out}}$.

4.3.5 Statistical Analysis

Table 4.2 indicates that models with four and five layers outperform those with more layers; however, there is no significant difference even between these two models. Aside from the high cost of deep learning models containing many layers, our findings indicate that the performance of 4-layer models is similar to that of models containing more layers, and we

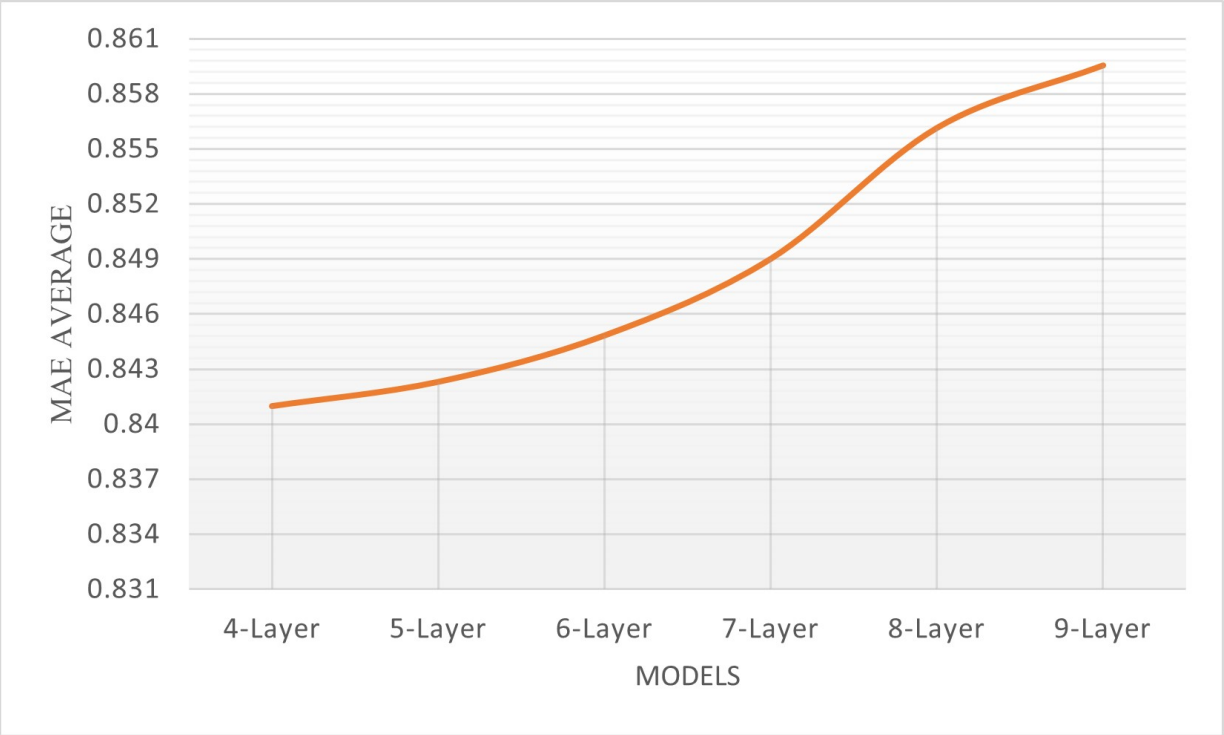

Figure 4.6: MAE across model depths (4 to 9 layers) after fine-tuning, with shallower models performing best.

conducted two independent statistical tests to prove this assertion. Table 4.3 summarizes the results of paired t-tests for 4-layer and 5-layer models before and after fine-tuning. The paired t-test [24] yielded p-values of 0.344 and 0.057, suggesting that there is no statistically significant difference in performance ($p > 0.05$) for both situations before and after fine-tuning. As illustrated in Table 4.4, we used the Wilcoxon signed-rank test [63] to validate our findings further, both before and after fine-tuning. Considering a significance level of 0.05 and a sample size of 12, the critical value is 13. Thus, the observed test statistics exceed the critical values, indicating that there is no significant difference in performance between the 4-layer and 5-layer models, both before and after fine-tuning. Considering these factors, using a deep learning model with four layers is recommended since it is simple and does not differ from those with more layers based on statistical tests.

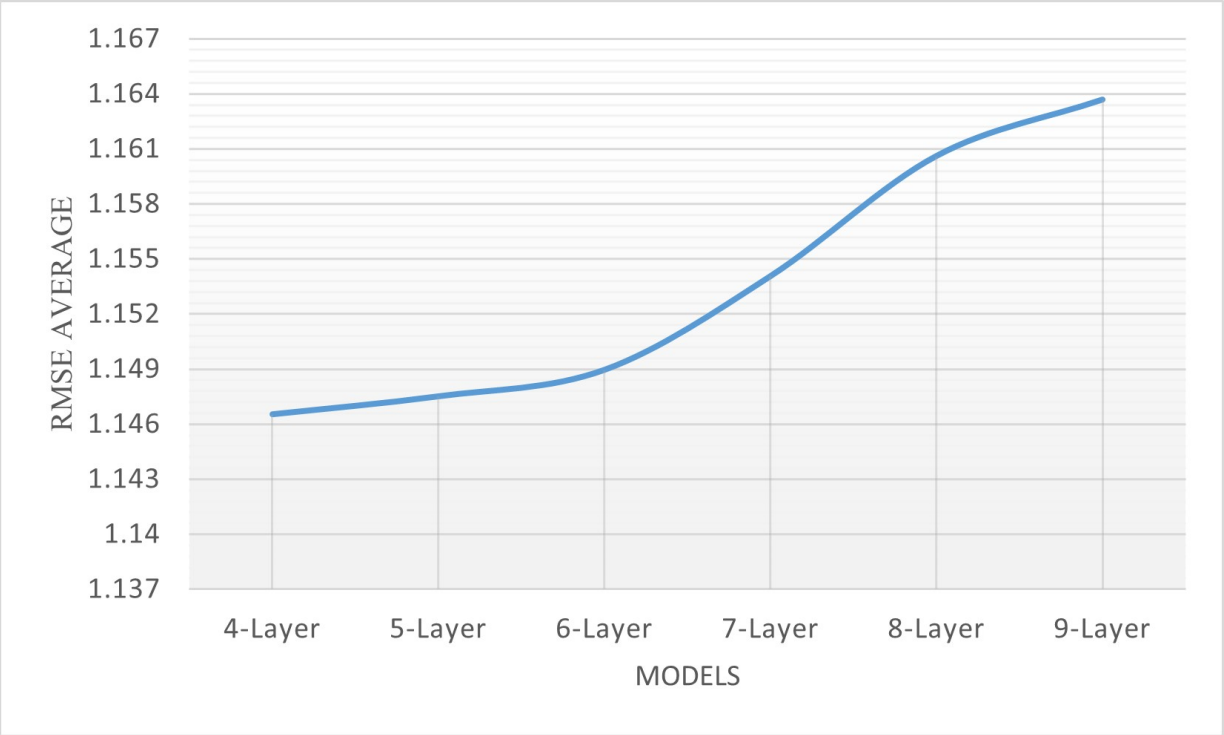

Figure 4.7: RMSE across model depths (4 to 9 layers) after fine-tuning, favoring fewer layers.

4.4 Discussion

Our research on deep learning architectures within the CDR landscape has yielded insights with practical implications. Contrary to prevailing assumptions, we reveal a non-linear relationship between layer depth and performance, which supports the choice of four layers. As we mentioned, current approaches using the rule of thumb by Blum [1] and the algebraic method by Cia et al. [3] do not have applications in deep learning models and can only be applied to a neural network with one hidden layer, as Omidvar et al. [49] proved that a 4-layer model outperformed a single-hidden-layer model. Furthermore, Boger et al.'s assertion [2] that the number of hidden nodes should be limited to capture up to 90% of the variance in the input dataset was found to lead to overfitting. This occurs when the

number of hidden nodes converges to this threshold, as demonstrated in Table 4.2 and Figure 4.3.

Our study findings contribute to both industry and academia by optimizing deep models to address specific problems they struggle with and build upon the theories. For industrial settings where efficiency and resource allocation are paramount, our findings offer a road map for developing a DNN architecture that meets performance demands within the confines of computational constraints. The time and space constraints that our algorithms need to follow can be efficiently managed by using techniques such as model quantization, pruning, and architecture search, which we addressed here. With the shift from a dogmatic "deeper is better" approach to a more detailed understanding of architectural complexity, organizations will be able to make informed decisions that align with their particular requirements, including the nature of their data, the intricacies of the task at hand, and the available resources.

In the academic context, our study adds a critical dimension to the ongoing discourse on deep learning model design and performance. The uncovered equilibrium between architectural complexity and accuracy challenges the traditional dogma that advocates for deep models without question. The findings of this study inspire researchers to examine in greater depth the underlying dynamics of simple architectures, resulting in a reevaluation of theoretical assumptions in the light of empirical evidence. In addition, the methodology used in this study includes empirical analysis, statistical testing, and evaluation metrics. The study's emphasis on balancing model performance with computational costs is consistent with ongoing initiatives to optimize deep learning solutions, demonstrating its usefulness as a reference for designing and optimizing deep learning algorithms. Thus, our research opens up future avenues of exploration as deep learning evolves; further studies could expand our findings to other domains and datasets, solidifying our work's robustness.

4.5 Summary

Even though deep learning has demonstrated remarkable performance in diverse domains, its performance is highly dependent on its structure. Considering this issue in the CDR, we investigated the performance of deep learning models with varying layer configurations, ranging from 4 to 9 layers. Although our analysis revealed that models with four and five layers consistently outperformed models with more layers, there was no significant difference in performance between these two models. The results suggest that simpler architectures can achieve comparable performance without incurring the computational costs of more complex models, and our statistical analysis confirmed these findings, suggesting that adding more layers beyond a certain point not only improves performance in our specific problem domain but also leads to a decline in performance. Concerning both computational efficiency and model performance, we find that a deep learning model with a specific number of layers strikes the most optimal balance between accuracy and computational cost.

Table 4.4: The result of the Wilcoxon signed rank test to determine the significant difference between the deep learning model with 4 and 5 layers before fine-tuning (BFT) and after fine-tuning (AFT) with $Diff$ denoting the difference between MAE or RMSE of two models and positive and negative ranks as W^+ and W^-, respectively.

BFT			**AFT**		
$Diff$	W^+	W^-	$Diff$	W^+	W^-
0.001333833	-	9	0.00090158	-	5
0.000993729	-	8	0.002399206	-	9
0.000250459	-	4	0.00405103	-	11
0.000109792	3	-	0.003300428	-	10
0.001354218	10	-	0.004074514	-	12
0.000346065	-	6	0.000252128	-	2
0.000108242	2	-	0.000332892	3	-
0.000422001	-	7	0.001274467	-	8
1.5378E-05	-	1	0.000971794	6	-
0.000269175	5	-	0.000338793	4	-
0.002834201	11	-	0.000199854	-	1
0.004097462	12	-	0.001011133	7	-
W Statistic	35		W Statistic	20	

Chapter 5

Conclusion and Future Works

5.1 Introduction

This chapter draws together the insights from the research and experiments conducted throughout this thesis. We start by emphasizing the contributions, implications of the results, and reflections on CDR systems and end by recognizing their limitations and possible future work. This research has contributed to more robust and accurate recommendation methodologies by confronting the challenges of the cold-start problem through deep personalized learning. In addition to summarizing the findings, this chapter identifies areas for future exploration to address unresolved challenges and unlock new opportunities in the field. The last chapter is organized into two main sections: the conclusion, which summarizes the primary findings and contributions of this work, and future works, which outline possible enhancements and extensions to refine and advance CDR systems further. Together, these sections provide a road map for applying the insights gained from this research, driving innovation and growth in intelligent recommendation systems.

5.2 Conclusion

This thesis uses a cross-domain approach to address the cold-start problem in recommender systems. The proposed model, DPTUPCDR, transfers user preferences from a source domain to a target one, aiming to improve recommendation accuracy for users with limited interaction history. Building upon the state-of-the-art model, it incorporates deeper neural networks and personalized bridge functions to capture better and transfer user preferences. Techniques such as attention mechanisms and meta-networks were also utilized to enhance feature selection and transferability.

We evaluated the model using the Amazon dataset, which covers categories such as movies, music, and books. Results showed significant improvements in key metrics, i.e., MAE, RMSE, Precision, Recall, and F1 Score. To ensure the model's stability and convergence, we carefully fine-tuned key hyperparameters and ran a set of statistical tests to ensure our model indeed performs better. These findings highlight the model's capability to tackle cold-start challenges effectively.

Although we tested the scalability and adaptability of the model by applying it to e-commerce and digital content consumption, there are limitations that will be discussed in the next section. During these tests, it was demonstrated that the model was able to maintain its effectiveness across different combinations of domains, resulting in reliable performance in a variety of contexts. Results show the importance of keeping the model as simple as possible, as this affects outcomes in real-world datasets. A systematic approach to optimizing hyperparameters and neural network depth further emphasizes the importance of configurations for achieving robust results.

The research not only contributes to CDR but also sheds light on the wider implications of TL frameworks. By creating bridges to transfer user preferences and find their tastes, the model solves the cold-start problem and paves the way for scalable and efficient recommender systems. Advanced neural techniques, including attention mechanisms, showcase the guarantee of these ideas in enhancing both quality and computational efficiency.

To summarize, this thesis improves recommender systems by presenting a cross-domain framework to tackle cold-start issues specifically. The findings stress the importance of using deep learning techniques to advance the state of the art in recommendation technologies.

5.3 Future Works

While this thesis presents a framework for addressing challenges associated with CDR systems, there are still opportunities for further improvements. In this section, we will explore several promising directions for enhancing the performance of the recommendation systems.

One potential direction involves developing a hybrid-based recommender system that integrates information from multiple source domains to improve the prediction accuracy in the final target domain. Drawing inspiration from systems like Netflix, which leverages CF by comparing user behaviors and CBF by analyzing item features, future studies could adopt hybridization techniques such as weighted combinations of component scores, switching among models based on context, or presenting mixed recommendations from various recommenders. Also, cascading models could prioritize specific algorithms while using others to refine results, and meta-level methods could generate an intermediate model as input for subsequent techniques. Such approaches, tailored to cross-domain settings, could average scores or construct sequential pipelines to refine recommendations progressively.

Another promising road lies in including implicit user feedback to mitigate data sparsity and improve recommendation performance in cold-start scenarios. In contrast to explicit ratings, which provide direct user preferences, implicit signals, such as click patterns, browsing behaviors, engagement times, and repeat interactions, are more available and do not bother to provide feedback all the time. Real-world systems naturally generate such feedback through interactions like clicks, purchases, and view times without requiring users to interrupt their experience to provide ratings or comments. For example, in elec-

tronic books, implicitly capturing user engagement, such as reading times and navigation patterns, can provide valuable insights without altering the natural reading behavior. Similarly, streaming platforms like Netflix and Amazon Prime benefit from implicit feedback by tracking the duration of watching time, which often provides a better understanding of preferences than explicit ones. Embedding these implicit behaviors into latent feature vectors, as reviewed in earlier chapters, can enable the model to reveal hidden patterns in user preferences that are not evident in the first place.

Alternative metrics in recommender systems go beyond accuracy measures like MAE and RMSE to capture other important aspects of recommendation quality. While Precision, Recall, and F1 Score provide insights into the relevance and coverage of recommendations, primarily when a classification-like approach is applied, they may not fully encompass the broader goals of recommendation quality. Diversity ensures recommendations are not repetitive, while coverage measures how much of the catalog gets explored, encouraging users to discover less popular items. Serendipity and novelty add an element of surprise and freshness by suggesting unexpected or previously unseen items that are still relevant. Mean Reciprocal Rank and Normalized Discounted Cumulative Gain focus on ranking quality and rewarding systems that prioritize the most relevant items at the top. By combining these metrics, future recommender systems will be able to provide users with an engaging, varied, and user-friendly experience that meets their needs.

Expanding the framework to include diverse datasets while integrating advanced NLP techniques can make CDR systems far more adaptable and effective. By working with datasets that reflect user behaviors and item types like healthcare records, academic resources, or niche platforms such as Goodreads or Spotify, models can learn to handle a broader range of data and adapt to new contexts. At the same time, applying NLP to analyze user feedback, such as reviews and comments, adds another layer of understanding. Techniques like sentiment analysis or powerful models like GPT and BERT can extract more profound insights into what users want, from their intent to emotional responses. Together, these approaches can strengthen the system's ability to provide meaningful recommendations.

By pursuing these directions, future work can build on the foundation established in this thesis to address current limitations and further enhance the effectiveness of CDR systems. These advancements will contribute to developing more adaptable, precise, and user-centric recommendation methodologies.

References

[1] Adam Blum. *Neural networks in C++: An object-oriented framework for building connectionist systems*. John Wiley & Sons, Inc., 1992.

[2] Zvi Boger and Hugo Guterman. Knowledge extraction from artificial neural network models. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 4, pages 3030–3035. IEEE, 1997.

[3] Guang-Wei Cai, Zhi Fang, and Yue-Feng Chen. Estimating the number of hidden nodes of the single-hidden-layer feedforward neural networks. In *2019 15th International Conference on Computational Intelligence and Security (CIS)*, pages 172–176. IEEE, 2019.

[4] Erion Çano and Maurizio Morisio. Hybrid recommender systems: A systematic literature review. *Intelligent data analysis*, 21(6):1487–1524, 2017.

[5] Devika Chhachhiya, Amita Sharma, and Manish Gupta. Designing optimal architecture of neural network with particle swarm optimization techniques specifically for educational dataset. In *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence*, pages 52–57. IEEE, 2017.

[6] Sanghyuk Roy Choi and Minhyeok Lee. Transformer architecture and attention mechanisms in genome data analysis: a comprehensive review. *Biology*, 12(7):1033, 2023.

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[8] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198, 2016.

[9] Yashar Deldjoo, Dietmar Jannach, Alejandro Bellogin, Alessandro Difonzo, and Dario Zanzonelli. Fairness in recommender systems: research landscape and future directions. *User Modeling and User-Adapted Interaction*, 34(1):59–108, 2024.

[10] Akbar Esfahanipour and Pouya Khodaee. A constrained portfolio selection model solved by particle swarm optimization under different risk measures. In *Applying Particle Swarm Optimization: New Solutions and Cases for Optimized Portfolios*, pages 133–153. Springer, 2021.

[11] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

[12] Chenjiao Feng, Jiye Liang, Peng Song, and Zhiqiang Wang. A fusion collaborative filtering method for sparse data in recommender systems. *Information Sciences*, 521:365–379, 2020.

[13] A Fiszelew, P Britos, A Ochoa, H Merlino, E Fernández, and R García-Martínez. Finding optimal neural network architecture using genetic algorithms. *Advances in Computer Science and Engineering Research in Computing Science*, 27:15–24, 2007.

[14] Wenjing Fu, Zhaohui Peng, Senzhang Wang, Yang Xu, and Jin Li. Deeply fusing reviews and contents for cold start users in cross-domain recommendation systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 94–101, 2019.

[15] Nadia Mushtaq Gardazi, Ali Daud, Muhammad Kamran Malik, Amal Bukhari, Tariq Alsahfi, and Bader Alshemaimri. Bert applications in natural language processing: a review. *Artificial Intelligence Review*, 58(6):1–49, 2025.

[16] Jyotirmoy Gope and Sanjay Kumar Jain. A survey on solving cold start problem in recommender systems. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 133–138. IEEE, 2017.

[17] Guibing Guo. Integrating trust and similarity to ameliorate the data sparsity and cold start for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 451–454, 2013.

[18] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention mechanisms in computer vision: A survey. *Computational visual media*, 8(3):331–368, 2022.

[19] Jaya Gupta, Sunil Pathak, and Gireesh Kumar. Deep learning (cnn) and transfer learning: a review. In *Journal of Physics: Conference Series*, volume 2273, page 012029. IOP Publishing, 2022.

[20] Abdul Mueed Hafiz, Shabir Ahmad Parah, and Rouf Ul Alam Bhat. Attention mechanisms and deep learning for machine vision: A survey of the state of the art. *arXiv preprint arXiv:2106.07550*, 2021.

[21] Saad Hikmat Haji and Adnan Mohsin Abdulazeez. Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 18(4):2715–2743, 2021.

[22] Yassine Himeur, Shahab Saquib Sohail, Faycal Bensaali, Abbes Amira, and Mamoun Alazab. Latest trends of security and privacy in recommender systems: a comprehensive review and future perspectives. *Computers & Security*, 118:102746, 2022.

[23] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2021.

[24] Henry Hsu and Peter A Lachenbruch. Paired t test. *Wiley StatsRef: Statistics Reference Online*, 2014.

[25] Guangneng Hu, Yu Zhang, and Qiang Yang. Conet: Collaborative cross networks for cross-domain recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 667–676, 2018.

[26] Kai Hu, Keer Xu, Qingfeng Xia, Mingyang Li, Zhiqiang Song, Lipeng Song, and Ning Sun. An overview: Attention mechanisms in multi-agent reinforcement learning. *Neurocomputing*, page 128015, 2024.

[27] Manisha Jangid and Rakesh Kumar. Deep learning approaches to address cold start and long tail challenges in recommendation systems: a systematic review. *Multimedia Tools and Applications*, pages 1–33, 2024.

[28] Umair Javed, Kamran Shaukat, Ibrahim A Hameed, Farhat Iqbal, Talha Mahboob Alam, and Suhuai Luo. A review of content-based and context-based recommendation systems. *International Journal of Emerging Technologies in Learning (iJET)*, 16(3):274–306, 2021.

[29] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost of deep learning models. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3873–3882. IEEE, 2018.

[30] Zahraa Saddi Kadhim, Hasanen S Abdullah, and Khalil I Ghathwan. Automatically avoiding overfitting in deep neural networks by using hyper-parameters optimization methods. *International Journal of Online & Biomedical Engineering*, 19(5), 2023.

[31] SeongKu Kang, Junyoung Hwang, Dongha Lee, and Hwanjo Yu. Semi-supervised learning for cross-domain recommendation to cold-start users. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1563–1572, 2019.

[32] Jaret M Karnuta, Sergio M Navarro, Heather S Haeberle, J Matthew Helm, Atul F Kamath, Jonathan L Schaffer, Viktor E Krebs, and Prem N Ramkumar. Predicting inpatient payments prior to lower extremity arthroplasty using deep learning: Which model architecture is best? *The Journal of Arthroplasty*, 34(10):2235–2241, 2019.

[33] Serhat Kılıçarslan, Kemal Adem, and Mete Çelik. An overview of the activation functions used in deep learning algorithms. *Journal of New Results in Science*, 10(3):75–88, 2021.

[34] 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.

[35] Hamidreza Koohi and Kourosh Kiani. Two new collaborative filtering approaches to solve the sparsity problem. *Cluster Computing*, 24:753–765, 2021.

[36] 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.

[37] Dokyun Lee and Kartik Hosanagar. Impact of recommender systems on sales volume and diversity. *N/A*, 2014.

[38] Minhyeok Lee. Mathematical analysis and performance evaluation of the gelu activation function in deep learning. *Journal of Mathematics*, 2023(1):4229924, 2023.

[39] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[40] Kangkang Li, Xiuze Zhou, Fan Lin, Wenhua Zeng, Beizhan Wang, and Gil Alterovitz. Sparse online collaborative filtering with dynamic regularization. *Information Sciences*, 505:535–548, 2019.

[41] Eva Lieskovská, Maroš Jakubec, Roman Jarina, and Michal Chmulík. A review on speech emotion recognition using deep learning and attention mechanism. *Electronics*, 10(10):1163, 2021.

[42] Alicia Lozano-Diez, Ruben Zazo, Doroteo T Toledano, and Joaquin Gonzalez-Rodriguez. An analysis of the influence of deep neural network (dnn) topology in bottleneck feature based language recognition. *PloS one*, 12(8):e0182580, 2017.

[43] Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. Cross-domain recommendation: An embedding and mapping approach. In *IJCAI*, volume 17, pages 2464–2470, 2017.

[44] Dionisis Margaris, Costas Vassilakis, and Dimitris Spiliotopoulos. On producing accurate rating predictions in sparse collaborative filtering datasets. *Information*, 13(6):302, 2022.

[45] Rachana Mehta and Keyur Rana. A review on matrix factorization techniques in recommender systems. In *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, pages 269–274. IEEE, 2017.

[46] Aanchal Mongia, Neha Jhamb, Emilie Chouzenoux, and Angshul Majumdar. Deep latent factor model for collaborative filtering. *Signal Processing*, 169:107366, 2020.

[47] Orly Moreno, Bracha Shapira, Lior Rokach, and Guy Shani. Talmud: Transfer learning for multiple domains. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 425–434, 2012.

[48] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[49] Sepehr Omidvar and Thomas Tran. Tackling cold-start with deep personalized transfer of user preferences for cross-domain recommendation. *International Journal of Data Science and Analytics*, pages 1–10, 2023.

[50] Weike Pan, Evan Xiang, Nathan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 230–235, 2010.

[51] Mohaimenul Azam Khan Raiaan, Sadman Sakib, Nur Mohammad Fahad, Abdullah Al Mamun, Md Anisur Rahman, Swakkhar Shatabda, and Md Saddam Hossain Mukta. A systematic review of hyperparameter optimization techniques in convolutional neural networks. *Decision analytics journal*, page 100470, 2024.

[52] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Introduction and challenges. *Recommender Systems Handbook*, pages 1–34, 2015.

[53] Laura Romero Meza and Giulio D'Urso. Navigating the paradox of choice in the digital age: A scoping review on the potential role of recommender systems. *World Futures*, pages 1–30, 2024.

[54] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, 2001.

[55] Badrul Sarwar, George Karypis, Joseph Konstan, and John T Riedl. Application of dimensionality reduction in recommender system-a case study. *None*, 2000.

[56] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):1–45, 2014.

[57] Nicollas Silva, Thiago Silva, Heitor Werneck, Leonardo Rocha, and Adriano Pereira. User cold-start problem in multi-armed bandits: When the first recommendations

guide the user's experience. *ACM Transactions on Recommender Systems*, 1(1):1–24, 2023.

[58] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[59] Derya Soydaner. Attention mechanism in neural networks: where it comes and where it goes. *Neural Computing and Applications*, 34(16):13371–13385, 2022.

[60] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. *Learning to Learn*, pages 3–17, 1998.

[61] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3:1–40, 2016.

[62] Christopher Wolters, Xiaoxuan Yang, Ulf Schlichtmann, and Toyotaro Suzumura. Memory is all you need: An overview of compute-in-memory architectures for accelerating large language model inference. *arXiv preprint arXiv:2406.08413*, 2024.

[63] Robert F Woolson. Wilcoxon signed-rank test. *Wiley Encyclopedia of Clinical Trials*, pages 1–3, 2007.

[64] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*, 2017.

[65] Xi Yang, Jie Yan, Wen Wang, Shaoyi Li, Bo Hu, and Jian Lin. Brain-inspired models for visual object recognition: an overview. *Artificial Intelligence Review*, 55(7):5263–5311, 2022.

[66] Junyong You, Guizhong Liu, and Hongliang Li. A novel attention model and its application in video analysis. *Applied mathematics and computation*, 185(2):963–975, 2007.

[67] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

[68] 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.

[69] Cheng Zhao, Chenliang Li, Rong Xiao, Hongbo Deng, and Aixin Sun. Catn: Cross-domain recommendation for cold-start users via aspect transfer network. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 229–238, 2020.

[70] Feng Zhu, Yan Wang, Chaochao Chen, Guanfeng Liu, Mehmet Orgun, and Jia Wu. A deep framework for cross-domain and cross-system recommendations. *arXiv preprint arXiv:2009.06215*, 2020.

[71] Feng Zhu, Yan Wang, Chaochao Chen, Jun Zhou, Longfei Li, and Guanfeng Liu. Cross-domain recommendation: Challenges, progress, and prospects. *arXiv preprint arXiv:2103.01696*, 2021.

[72] Yongchun Zhu, Kaikai Ge, Fuzhen Zhuang, Ruobing Xie, Dongbo Xi, Xu Zhang, Leyu Lin, and Qing He. Transfer-meta framework for cross-domain recommendation to cold-start users. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1813–1817, 2021.

[73] Yongchun Zhu, Zhenwei Tang, Yudan Liu, Fuzhen Zhuang, Ruobing Xie, Xu Zhang, Leyu Lin, and Qing He. Personalized transfer of user preferences for cross-domain recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1507–1515, 2022.

APPENDICES

The following appendices contain the pseudocode and supplementary materials for this thesis. Each appendix corresponds to an essential component of the DPTUPCDR framework, detailing the preprocessing steps, model architecture, and runner module. These pseudocode representations are included to enhance clarity, ensure reproducibility, and provide a thorough understanding of the methods used in this work. For the exact implementation and additional resources, refer to the project's GitHub repository https://github.com/SepehrOmidvar/DPTUPCDR.

Appendix A

Preprocessing

The preprocessing step prepares the datasets for training, validation, and testing in the DPTUPCDR framework. It consists of two main stages:

- Mid-Level Preprocessing: Converts raw JSON data into a structured CSV format containing user IDs (uid), item IDs (iid), and ratings (y). This step ensures compatibility with the model pipeline by standardizing input formats across domains.
- Ready-Level Preprocessing:
    1. Reads the processed mid-level CSV files for the source and target domains.
    2. Maps user and item IDs to unified indices to facilitate cross-domain training.
    3. Splits the data into training and testing datasets based on a predefined ratio.
    4. For overlapping users in the source and target domains, it constructs positive interaction sequences to capture historical behaviors.
    5. Saves the prepared datasets for use in the training and evaluation phases.

Here is the pseudocode:

```
INPUT: Raw JSON data files for each domain
OUTPUT: Split datasets: 'train_src.csv', 'train_tgt.csv', 'train_meta.csv', 'test.csv'

# Mid-Level Preprocessing
FOR each domain IN Domains:
    OPEN 'reviews_{domain}_5.json.gz'
    RE = []
    FOR each line IN file:
        PARSE JSON line
        uid = line['reviewerID']
        iid = line['asin']
        y = line['overall']
        APPEND [uid, iid, y] TO RE
    SAVE RE TO 'mid/{domain}.csv'

# Ready-Level Preprocessing
src = READ 'mid/{src}.csv'
tgt = READ 'mid/{tgt}.csv'

# Map IDs to unified indices
uid_dict = CREATE_MAPPING(UNION(UNIQUE(src.uid), UNIQUE(tgt.uid)))
iid_dict_src = CREATE_MAPPING(UNIQUE(src.iid))
iid_dict_tgt = CREATE_MAPPING(UNIQUE(tgt.iid), OFFSET=len(iid_dict_src))

src.uid, src.iid = MAP(src.uid, uid_dict), MAP(src.iid, iid_dict_src)
tgt.uid, tgt.iid = MAP(tgt.uid, uid_dict), MAP(tgt.iid, iid_dict_tgt)

# Split data
co_users = INTERSECTION(UNIQUE(src.uid), UNIQUE(tgt.uid))
```

```
test_users = RANDOM_SAMPLE(co_users, SIZE=RATIO[1] * len(co_users))
train_src = src
train_tgt = FILTER(tgt, uid NOT IN test_users)
train_meta = FILTER(tgt, uid IN (co_users - test_users))
test = FILTER(tgt, uid IN test_users)

# Add positive interaction sequences
pos_seq_dict = {uid: GET_POSITIVE_ITEMS(src, uid) FOR uid IN co_users}
train_meta['pos_seq'] = MAP(train_meta['uid'], pos_seq_dict)
test['pos_seq'] = MAP(test['uid'], pos_seq_dict)

# Save split datasets
SAVE train_src TO 'ready/train_src.csv'
SAVE train_tgt TO 'ready/train_tgt.csv'
SAVE train_meta TO 'ready/train_meta.csv'
SAVE test TO 'ready/test.csv'
```

Appendix B

Model

The model architecture consists of three key components:

- LookupEmbedding: Generates embeddings for user and item IDs, which are the foundational representations used in all stages of training and evaluation.
- DeepMetaNet: Implements an attention mechanism to aggregate user interaction histories and generates transformation matrices for domain mapping.
- Combines the functionalities of LookupEmbedding and DeepMetaNet to support different training stages.

Here is the pseudocode:

```
CLASS LookupEmbedding:
    INIT: Define embedding layers for user IDs and item IDs
    FUNCTION forward(x):
        RETURN concatenated embeddings for user and item IDs

CLASS DeepMetaNet:
```

```
    INIT:
        DEFINE attention mechanism to aggregate user interaction sequences
        DEFINE multi-layer decoder to compute transformation matrices
    FUNCTION forward(emb_fea, seq_index):
        Compute attention scores for user-item interaction sequences
        Aggregate features using attention weights
        RETURN transformation matrix for embedding alignment

CLASS Model:
    INIT:
        DEFINE LookupEmbedding for source, target, and augmented domains
        DEFINE DeepMetaNet for user preference transfer
        DEFINE bridge functions (linear layers) for personalized embedding alignment
    FUNCTION forward(x, stage):
        IF stage == 'train_src' OR 'train_tgt' OR 'train_aug':
            RETURN interaction scores using user-item embeddings
        ELSE IF stage == 'train_meta':
            Generate transformation matrix using DeepMetaNet
            Align user embeddings from source to target domain
            RETURN interaction scores with aligned embeddings
        ELSE IF stage == 'train_map':
            RETURN aligned embeddings from personalized bridge functions
        ELSE IF stage == 'test_map':
            Apply embedding alignment and RETURN interaction scores
```

Appendix C

Runner

The runner module integrates the preprocessing and model components, executing the training, evaluation, and optimization steps of the DPTUPCDR framework. It initializes the model, trains it across various stages, evaluates performance using MAE and RMSE metrics, and logs the results for analysis. Here is the pseudocode:

```
IMPORT Preprocessing modules (Mid-Level and Ready-Level)
IMPORT Model modules (LookupEmbedding, DeepMetaNet, and Model)

INPUT: Configuration file, training parameters, and dataset paths
OUTPUT: Trained model and evaluation metrics

# Step 1: Configuration and Initialization
LOAD configuration file
SET random seed for reproducibility
INITIALIZE Model with embedding layers, meta-network, and bridge functions
DEFINE loss function (e.g., MSE)
DEFINE optimizers and learning rate schedulers
```

```
# Step 2: Training
FOR each stage IN ['train_src', 'train_tgt', 'train_aug', 'train_meta', 'train_map']:
    FOR each epoch:
        TRAIN model on the corresponding dataset
        UPDATE model parameters using optimizer and scheduler

# Step 3: Evaluation
FOR each stage IN ['test_tgt', 'test_aug', 'test_meta', 'test_map']:
    EVALUATE model on the test dataset
    CALCULATE metrics (MAE, RMSE, Precision, Recall, F1 Score)
    UPDATE best metrics if improved

# Step 4: Results Logging
LOG final metrics for all stages
SAVE trained model and evaluation results
```