UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Trustworthy Recommender Systems: Robustness and Privacy Perspectives

by Shanshan Ye

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy in Computer Science

under the supervision of A/Prof. Guangquan Zhang, Distinguished Prof. Jie Lu, and Dr. Qian Zhang

University of Technology Sydney
Faculty of Engineering and Information Technology

January 2026

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Shanshan Ye*, declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *School Computer Science*, *Faculty of Engineering and Information Technology* at the University of Technology Sydney, Australia. This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution. This research was supported by an Australian Government Research Training Program (RTP) Scholarship doi.org/10.82133/C42F-K220.

SIGNATURE: Production Note:
Signature removed prior to publication.

[Shanshan Ye]

DATE: January, 2026

PLACE: Sydney, Australia

Abstract

Trustworthy recommender systems are essential for providing personalised, reliable, and secure services in the face of increasing data complexity and stringent privacy regulations. While traditional systems rely on high-quality user-item interactions, real-world scenarios often involve "low-quality" input data, e.g., characterised by limited feedback and corruption, and rising demands for data sovereignty through "right to be forgotten" mandates. This thesis addresses these critical challenges by proposing a comprehensive framework for building trustworthy recommender systems that ensure reliability under data adversity and privacy-preserving requirements.

The first half of this research focuses on improving reliability when training data is imperfect. We first address the cold-start and limited-feedback problem by investigating a setting in which only browsing histories are available, without explicit acquisition signals (e.g., purchases or ratings). We propose the Partial Acquisition Recommender System (PARS), which draws an analogy to partial label learning to extract reliable preferences from ambiguous sequences. We then address rating flip noise, a prevalent issue in which interaction history is corrupted. By modelling the noise generation process through a noise transition matrix, we develop a statistically consistent algorithm that enables the system to remain robust against corrupted ratings, significantly outperforming traditional models that assume noise-free data.

The second half of the thesis shifts toward privacy preservation through machine unlearning. As users increasingly demand the removal of their sensitive information,

retraining models from scratch becomes computationally prohibitive. We introduce a novel targeted label noise injection method that allows for the efficient removal of specific client data without accessing the remaining training set, a critical requirement when data is unavailable due to expiration or storage limits. This approach also encourages the model to lose confidence in "to-be-forgotten" sequences by moving them across the decision boundary with minimal impact on overall recommendation accuracy. We further extend this framework to robust sequential unlearning, ensuring that the "right to be forgotten" is upheld even in complex sequential architectures without sacrificing generalisation performance.

Through extensive empirical evaluations on real-world benchmarks, this thesis demonstrates that trustworthy recommender systems can be achieved by balancing predictive power with data integrity and user privacy. By mitigating the risks of system bias, noise, and privacy leaks, this research contributes to the long-term viability and ethical deployment of recommendation technologies across diverse digital domains.

Dedication

To myself and my family...

Acknowledgments

Going through a PhD study has been a challenging, exciting and unforgettable journey, and it is one of the paths that has changed the trajectory of my life. I am deeply grateful for and cherish this experience; although it lasted only three and a half years, the impact and meaning it has had on me are profound and enduring. Here, I would like to take this opportunity to express my sincere appreciation to all those who have accompanied, encouraged, mentored, and supported me throughout this journey.

First, I would like to express my heartfelt gratitude to my supervisors, Distinguished Professor Jie Lu, Associate Professor Guangquan Zhang and Dr Qian Zhang, for their generous guidance and support. Their expertise and mentorship were instrumental throughout my PhD journey, and this work would not have been possible without their help. Distinguished Prof. Jie Lu is not only a supervisor but also a role model. In terms of research, she has consistently provided professional and effective advice that helps me avoid unnecessary detours in my academic journey. Beyond academia, she embodies strength and women's power. Her wisdom, kindness and the values she brings to society and others have deeply inspired me. She is also like a friend, approachable and easygoing, always willing to share thoughtful advice and practical support for challenges encountered in life. A/Prof. Guangquan Zhang provided highly constructive advice on the direction of my research. His firm grasp of the broader picture, along with his professionalism and efficiency in academic research, has been of great help to me. He

has always encouraged me to explore more innovative ideas and go deeper beyond those initiatives. Dr Qian Zhang gave me detailed, critical insights into conducting research effectively and writing strong research papers. Her patience and instructive feedback has been of great help to me.

I am especially grateful to the University of Technology Sydney for supporting my PhD research through the Faculty of Engineering Research Scholarship.

My sincere appreciation also goes to the team members in the Decision Systems and e-Service Intelligence (DeSI) lab at the Australian Artificial Intelligence Institute (AAII). Being part of this fabulous group has been a truly delightful and rewarding experience, from which I have gained friendship, support, knowledge, inspiration and encouragement. In particular, I would like to thank Dr Zhen Fang for his invaluable assistance and guidance in my research; he offered many constructive suggestions to improve my research techniques and skills. I am also grateful to Dr Kezhi Lu for the meaningful research discussions and experimental collaboration we shared, to Dr Pham Minh Thu Do for insightful discussions that brought new information and ideas, and to Dr Kuo Shi for his help and support in my research work.

Finally, I would like to express my heartfelt gratitude to my family. My husband, who loves and supports me unconditionally, gave me the courage to embark on this journey. My son, whose innocent loveliness and sweetness have always eased the pressure and busyness of my research life. I am also deeply thankful to my mother and my mother-in-law, who were willing to take turns coming to Sydney to help us manage household responsibilities and care for our child, allowing me to devote myself entirely to my research. I feel truly fortunate and grateful to have them in my life; their sacrifice and constant behind-the-scenes support have enabled me to complete this PhD journey with joy and fulfillment.

List of Publications

Papers included in the thesis:

1. **Shanshan Ye**, Kezhi Lu, Guangquan Zhang, and Jie Lu. "PARS: Partial-Label-Learning-inspired Recommender Systems." In Proceedings of the AAAI Conference on Artificial Intelligence. 2025. **[Chapter 3]**

2. **Shanshan Ye** and Jie Lu. "Robust recommender systems with rating flip noise." ACM Transactions on Intelligent Systems and Technology 16, no. 1 (2024): 1-19. **[Chapter 4]**

3. **Shanshan Ye** and Jie Lu. "Sequence unlearning for sequential recommender systems." In Australasian Joint Conference on Artificial Intelligence, pp. 403-415. Singapore: Springer Nature Singapore, 2023. **[Chapter 5]**

4. **Shanshan Ye**, Jie Lu, and Guangquan Zhang. "Towards safe machine unlearning: A paradigm that mitigates performance degradation." In Proceedings of the ACM on Web Conference 2025, pp. 4635-4652. 2025. **[Chapter 6]**

Table of Contents

List of Figures

Figure **Page**

List of Tables

Table **Page**

CHAPTER 1

INTRODUCTION

1.1 Introduction and Background

We are firmly established in the era of big data. The past few decades have witnessed a remarkable surge in the resources available due to the wave of digitalisation, creating a digital landscape of unprecedented scale. E-commerce giants such as Amazon, Alibaba, and eBay manage millions of users and hundreds of millions of products across nearly every conceivable category. Simultaneously, multimedia streaming providers like Spotify and Netflix offer access to exhaustive catalogues of music and movies, while short video platforms, including TikTok, YouTube Shorts, and Instagram Reels, generate millions of videos daily, each catering to diverse and niche aesthetic preferences.

While this abundance offers significant choice, the sheer volume of information has become increasingly overwhelming. For the end-user, obtaining relevant information from these vast collections is often difficult, time-consuming, and cognitively exhausting. Consequently, recommender systems (RS) have become indispensable tools designed to improve user experience by streamlining information acquisition and task retrieval. By

analysing historical interactions, these systems predict individual preferences to surface the most relevant items, creating a value chain that benefits business, education, culture, and entertainment services alike (Davidson et al., 2010).

The role of recommender systems has undergone a significant transformation since their inception. Initially utilised by early e-commerce pioneers to suggest related books or electronics, these systems are now a foundational component of the global digital infrastructure. They have fundamentally altered how business is conducted, expanding far beyond retail into sectors such as government services, digital libraries, social media, and the tourism industry (Lu et al., 2020). This expansion has been fueled by a shift in technical complexity. We have moved from basic collaborative filtering, which relied on simple user-item grids, to deep learning architectures and foundation models that process multi-modal data (text, images, and video). For industry leaders like TikTok, Netflix, YouTube, and Amazon, these sophisticated engines are no longer just "features"; they are the primary drivers of user retention and operational success (Lu et al., 2015).

However, as recommender systems have grown more complex and data-dependent, a critical tension has emerged between the theoretical "clean" data used in research and the "messy" reality of industrial data streams. Despite the successful development of recommendation systems over the last decade, real-world big data is frequently characterised by imperfect information. In the pursuit of higher accuracy, systems must now contend with data that is missing, corrupted, or sensitive. This "low-quality" nature manifests in several ways, i.e., a user is likely to only review products without any feedback information; a user may enjoy only a specific function of a product, yet the system records an ambiguous rating. A user may randomly click ratings or provide feedback without careful reflection, corrupting the interaction history.

Moreover, deep learning and foundation models have demonstrated remarkable performance across a wide range of tasks. They provide excellent opportunities for

building powerful recommender systems (Cui et al., 2022). We believe that large model- and foundation-model-based recommender systems will be popular in the near future. However, large models have a very large number of parameters, making it easy to remember the data. For example, it is reported that diffusion models memorise individual images from their training data and emit them at generation time (Carlini et al., 2023). How to protect the privacy of users is essential when designing reliable recommender systems, which is a foundation for us to deploy recommender systems widely.

Historically, the primary metric for evaluating recommender systems has been predictive accuracy. In the current era of big data, the path to further enhancing this accuracy lies in the ability to effectively exploit the massive volume of available "imperfect" data. Rather than discarding or ignoring interactions characterised by missing feedback or noisy attributes, building recommender systems robust enough to safely extract and utilise this invaluable information is the key to pushing the boundaries of recommendation performance.

Moreover, as recommender systems become more accurate and data-intensive, it has become clear that accuracy alone is no longer a sufficient measure of success. We must also prioritise user privacy and data sovereignty. The emergence of strict privacy regulations and the "right to be forgotten" presents a significant technical hurdle for modern recommender systems. Consequently, the challenge of the next generation of recommender systems is to maintain high accuracy while ensuring the system remains trustworthy and privacy-compliant.

These challenges have shifted the research focus toward developing trustworthy recommender systems. While the academic community recognises five primary pillars of trustworthiness, i.e., robustness, privacy, fairness, explainability, and transparency, this thesis identifies robustness and privacy as the most urgent foundations for modern deployment (Kaur et al., 2022). The thesis focuses on these two pillars, aiming to bridge

the gap between theoretical models and real-world requirements. By addressing the lack of explicit feedback and corrupted interaction histories, and by developing robust unlearning algorithms, this research contributes to the creation of recommender systems that are not only powerful but inherently reliable and privacy-preserving.

1.2 Research Questions and Objectives

1.2.1 Research Questions

Data imperfection continues to pose significant challenges for recommender systems. To address this issue, this study investigates the following research questions toward building robust recommender systems:

RESEARCH QUESTION 1 (RQ1): *How to build robust recommender systems without explicit feedback?*

Recommender systems are widely used to solve many real-world problems. To achieve strong performance in practical applications, recommender systems must accurately capture user interests. Traditionally, recommender systems rely heavily on explicit feedback, such as users' ratings, preferences, or purchase records, to construct effective recommendation models. However, in many real-world scenarios, explicit feedback is often unavailable due to data accessibility limitations or privacy concerns. For instance, users may browse product pages, read articles, or watch videos without providing ratings, expressing preferences, or completing purchases. In these settings, where only implicit signals such as browsing histories are available, building effective recommender systems becomes both challenging and essential.

RESEARCH QUESTION 2 (RQ2): *How to deal with noisy feedback when building recommender systems?*

The performance of recommender systems largely depends on user-item interaction

histories, which are assumed to reflect users' preferences accurately. In practice, however, this assumption is often violated because interaction data can be noisy or corrupted. As a result, recommender systems built on such data may become unreliable and untrustworthy. Specifically, due to the inherent imprecision of the ratings, e.g., randomness on the clicking, many ratings are usually contained with noise. For example, some may just randomly click any ratings from 0-9 after watching or even without watching a movie. Those noises will degenerate the existing recommender systems, especially those based on deep learning because deep networks can easily overfit into any noise. Therefore, how to build robust recommender systems is crucial by exploiting these noisy data without reflecting users' true preferences.

RESEARCH QUESTION 3 (RQ3): *How to efficiently remove client information from sequential recommender systems to protect their privacy?*

It is essential to protect users' personal information. The USA and the UK have different legal frameworks to protect personal data and information. For example, the USA has the California Consumer Privacy Act (CCPA), which grants California residents certain rights regarding their personal data and imposes obligations on businesses that collect or sell such data. It allows individuals to be aware of what personal information is being collected and shared, the right to opt out of the sale of their data, and the right to request the deletion of their data. The UK has the Data Protection Act 2018, which implements the General Data Protection Regulation (GDPR) in the UK. It sets out the rules for processing personal data, including its collection, utilisation, storage, and sharing. It provides individuals with rights over their data and imposes obligations on organisations to handle data responsibly. Protecting users' personal information is becoming increasingly important as large models are thriving. OpenAI is now looking for teams worldwide to expand proofs-of-concept for a democratic process that could solve the inquiries about what rules AI systems should follow, e.g., to project and legitimately

use personal data. It will become natural for a user to request a company to remove their personal information from the use of training AI models. But how can we handle this data removal request efficiently without costly and time-consuming as an expense is really essential.

RESEARCH QUESTION 4 (RQ4): *How to effectively unlearn specific training data from a pre-trained recommender model while maintaining the model's performance?*

Machine unlearning, which aims to remove specific data from trained machine learning models, has become an important capability for ensuring compliance with data privacy regulations and maintaining user trust. However, the unlearning process often leads to a decline in model performance. To alleviate this issue, traditional approaches usually rely on access to both the data that needs to be removed and the remaining training data, using the retained data to fine-tune the model after unlearning. In practice, however, access to the remaining data is often infeasible due to factors such as data expiration policies, storage constraints, or privacy restrictions that limit data reuse without explicit consent. As a result, performing effective unlearning without access to the remaining data presents substantial challenges. My research will tackle the issue of performance drop while doing the machine unlearning.

1.2.2 Research Objectives

To answer these research questions, we set up the corresponding Research Objectives (RO) as follows:

RESEARCH OBJECTIVE 1 (RO1): *To build robust recommender systems rely only on browsing histories* (Aims to answer RQ1)

Having users' explicit feedback information is important to build effective recommender systems. In order to recover their feedback, we draw an analogy of problem setting to partial Label Learning. The setting of Partial label learning is quite similar to

our problem setting. Our method is to use the idea of partial label learning to recover the explicit user feedback. However, to do so, we have to make an assumption: the majority of sequences contain at least one item purchased. The experiment we conducted shows that the assumption is not strong. Therefore, we can confidently apply the methodology of partial label learning to our problem so as to recover users' explicit feedback relying only on browsing histories.

RESEARCH OBJECTIVE 2 (RO2): *To build robust recommender systems from noisy ratings.* (Aims to answer RQ2)

Most approaches of recommender systems are based on the assumption that the interaction between users and items is reliable and accurate. However, in a real-life scenario, the assumption may not always hold true, where the interaction data can be noisy and unreliable. We propose two methods to solve this problem: a) extract confident ratings from the noisy ones; b) link the observed noisy ratings with the latent correct ratings by using a probabilistic model, which is also called a transition model. The key of our proposed mechanism is to learn the transition matrix from the given data. Intuitively, by acquiring the relationship between clean ratings and flipped ratings and the noisy ratings, we can refer to the clean rating information and thereby build robust recommender systems.

RESEARCH OBJECTIVE 3 (RO3): *To protect users' private information by exploiting user information unlearning techniques for recommender systems.* (Aims to answer RQ3)

Unlike the traditional approach of removing users' personal data and retraining the model, which is costly and time-consuming. We propose a novel sequence unlearning method by injecting label noise into the original label sequence. The intuition of our approach is to make the model forget certain sequences by encouraging the system to produce random predictions for them. To prevent the model from overfitting to an incorrect

label, which could cause large parameter changes, we have designed a dynamic process where the incorrect label is continually altered during the training phase. This effectively facilitates the model to lose confidence in the original label, while also preventing it from overfitting to a specific wrong label.

RESEARCH OBJECTIVE 4 (RO4): *To develop safe machine unlearning techniques for recommender systems without accessing the remaining training data and protect performance without dramatically changing the model's parameters.* (Aims to answer RQ4)

Conventional machine unlearning methods often affect model performance after the unlearning process. To address this issue, most existing approaches rely on access to the remaining training data to fine-tune the model and lessen the impact of unlearning. In practice, however, accessing the remaining data is not always feasible due to factors such as data expiration policies, storage limitations, or privacy restrictions. As a result, performing machine unlearning without access to the remaining training data while maintaining model performance is highly challenging. Therefore, we investigate how to remove specific training data from a pre-trained model without using the remaining training data and without significantly altering the model's parameters. We propose an effective approach called targeted label noise injection. The core idea is to assign incorrect but carefully controlled labels to the data that needs to be forgotten and then fine-tune the model on these modified labels. This approach effectively shifts the to-be-forgotten examples across the decision boundary while having minimal impact on the model's overall performance.

1.3 Research Contributions

This thesis shows a comprehensive analysis of various data exploitation situations in different real-life scenarios and reveal the main challenges exposed when building

trustworthy recommender systems under respective situations. Correspondingly, a series of approaches has been proposed to address these challenges. The main contributions of this study are summarised as follows:

1) A new partial-label-learning-inspired framework for browsing-only recommender systems.

- This study proposes a novel Partial Acquisition Recommender System (PARS) for sequential recommendation in scenarios where only user browsing histories are available, addressing the lack of explicit acquisition feedback under realistic data accessibility and privacy constraints.
- A unified learning framework is developed by integrating masked self-supervised sequence modelling with partial label learning, enabling the extraction and progressive refinement of latent item acquisition information from unlabelled browsing sequences.
- Extensive experiments on multiple real-world benchmark datasets are conducted to evaluate the proposed framework, demonstrating its effectiveness, robustness, and practical relevance, particularly in cold-start and privacy-preserving recommendation settings.

2) A novel framework for learning robust recommender systems under rating flip noise.

- This study investigates a realistic and challenging form of corrupted interaction history in recommender systems, namely rating flip noise, which captures a wide range of unreliable, adversarial, and randomly generated rating behaviours beyond commonly assumed additive noise.
- A noise-aware recommendation framework is introduced by modelling the rating flip process through a noise transition matrix, transforming the recommendation

task into a multi-class classification problem and enabling principled inference of clean rating information from noisy observations.

- A statistically consistent learning strategy is developed by constructing an unbiased estimator of the noise transition matrix using anchor points, leading to robust recommender systems whose effectiveness and superiority are validated through comprehensive experiments on multiple benchmark datasets.

3) A practical unlearning framework for sequential recommender systems without retraining.

- This study proposes a novel sequence unlearning method for sequential recommender systems based on label noise injection, addressing the challenge of removing specific client information under privacy regulations while avoiding costly model retraining.

- A targeted fine-tuning strategy is introduced to promote unlearning by encouraging random predictions on the sequences to be forgotten, effectively emulating the behaviour of a model retrained on the remaining data while limiting changes to model parameters.

- A dynamic label perturbation mechanism is designed to prevent overfitting to incorrect labels during unlearning, ensuring that the model loses confidence in forgotten sequences while maintaining strong generalisation performance on the remaining data across different datasets.

4) A new and safe machine unlearning mechanism that mitigates performance degradation for recommender systems

- This study proposes a novel machine unlearning method, termed targeted label noise injection, to safely remove specific training examples from a pre-trained

model, addressing privacy requirements while avoiding the heavy performance degradation commonly observed in existing approaches.

- A principled unlearning strategy is introduced by assigning incorrect yet easy-to-learn labels to the to-be-forgotten examples, enabling the model to shift these examples across the decision boundary while minimising parameter changes and preserving performance on the remaining data.

- A controlled loss modulation mechanism is designed to prevent overfitting to pseudo-labels during fine-tuning, ensuring that parameter updates remain localised. Theoretical analysis and extensive experiments demonstrate that the proposed method achieves state-of-the-art unlearning performance with minimal impact on model utility across diverse datasets.

1.4 Research Significance

The theoretical and practical significance of this thesis is summarised as follows:

Theoretical significance: This thesis systematically investigates the problem of building trustworthy recommender systems under a range of realistic data challenges, including missing explicit feedback, corrupted interaction histories, and data removal requirements driven by privacy regulations. By relaxing several strong assumptions commonly adopted in existing recommender system research, such as the availability of reliable acquisition feedback, the correctness of user-item interactions, and unrestricted access to training data, this work contributes to a broader and more realistic theoretical understanding of recommendation learning.

Specifically, this thesis establishes principled learning frameworks for partial-label-learning-based recommendation, robust recommendation under rating flip noise, and machine unlearning without access to remaining training data. In addition, it contributes

a theoretical understanding of safe unlearning, where user data can be removed while explicitly controlling parameter updates and preventing unnecessary performance degradation. By linking recommender systems with partial label learning, noisy label learning, concept unlearning and robust unlearning, this work provides new insights into robustness, stability, and controllability in learning from imperfect and evolving data. These contributions enrich the theoretical foundations of recommender systems and open new research directions for privacy-aware and reliability-driven learning.

Practical significance:

The findings of this thesis are highly relevant to real-world recommender systems that must operate under privacy regulations, data quality issues, and limited access to training data. The proposed methods enable effective recommendation using only browsing histories when explicit feedback is unavailable, improve resilience against unreliable and adversarial rating behaviours, and support efficient unlearning of user data without requiring costly retraining or access to remaining datasets.

Importantly, this thesis demonstrates that user data can be safely removed with minimal impact on model performance through carefully designed unlearning mechanisms that limit parameter changes. Extensive experimental evaluations on multiple real-world benchmark datasets show that the proposed approaches achieve strong generalisation performance while fulfilling data deletion requests. Together, these results provide practical guidance for deploying robust, privacy-preserving, and trustworthy recommender systems in modern online environments, and offer insights applicable to a broader range of data-driven applications.

1.5 Thesis Structure

The structure of the thesis is shown in Figure 1.1 and the chapters are organised as follows:

- CHAPTER 2: This chapter provides a comprehensive literature review related to this research. It begins by introducing the fundamental concepts and types of recommender systems. Afterwards, it illustrates the basic processes and mainstream methods for recommender systems. Furthermore, it shows the other aspects of recommender systems, such as cross-domain recommender systems and robust recommender systems with noisy data.

- CHAPTER 3: This chapter shows our accepted conference paper" PARS: Partial-Label-Learning-Inspired Recommender Systems". It introduces methods for building robust recommender systems in settings with lack of explicit feedback. We propose a method called Partial Acquisition Recommender Systems (PARS), where we draw an analogy to the setting of partial label learning in weakly supervised learning and enables us to train reliable recommender systems only using browsing histories. Empirical results on popular real-world benchmark datasets consistently show the effectiveness of our approach PARS. This chapter addresses RQ1 to achieve RO1.

- CHAPTER 4: This chapter presents our accepted journal paper "Robust Recommender Systems with Rating Flip Noise". It first elaborates on how rating-flip noise affects recommender system performance and reviews related work on recommendation models. Subsequently, we examine how to handle this type of noise and propose utilising the noise transition matrix to model it. We conduct extensive experiments on the datasets corrupted by rating-flipped noise. This chapter addresses RQ2 to achieve RO2.

- CHAPTER 5: This chapter shows our accepted conference paper "Sequence unlearning for sequential recommender systems". It focuses on how to efficiently remove specific client information from a pre-trained sequential recommender system without retraining, particularly when the dataset change is not substantial. We

propose a novel sequence unlearning method for sequential recommender systems by leveraging label noise injection. Empirically, we show that our method enables different recommender systems to forget specific sequential data while maintaining strong generalisation performance on the remaining data across multiple datasets. This chapter addresses RQ3 to achieve RO3.

- CHAPTER 6: This chapter presents our accepted conference paper "Towards Safe Machine Unlearning: A Paradigm that Mitigates Performance Degradation". We study how to unlearn specific training data from a pre-trained model without accessing the remaining training data, while preserving model performance without dramatically changing the model's parameters. We propose a practical method called Targeted Label Noise Injection. We theoretically prove the effectiveness of the proposed method and empirically show that it achieves state-of-the-art unlearning performance across various datasets. This chapter addresses RQ4 to achieve RO4.

- CHAPTER 7 summarises the main findings of this thesis and outlines directions for future research.

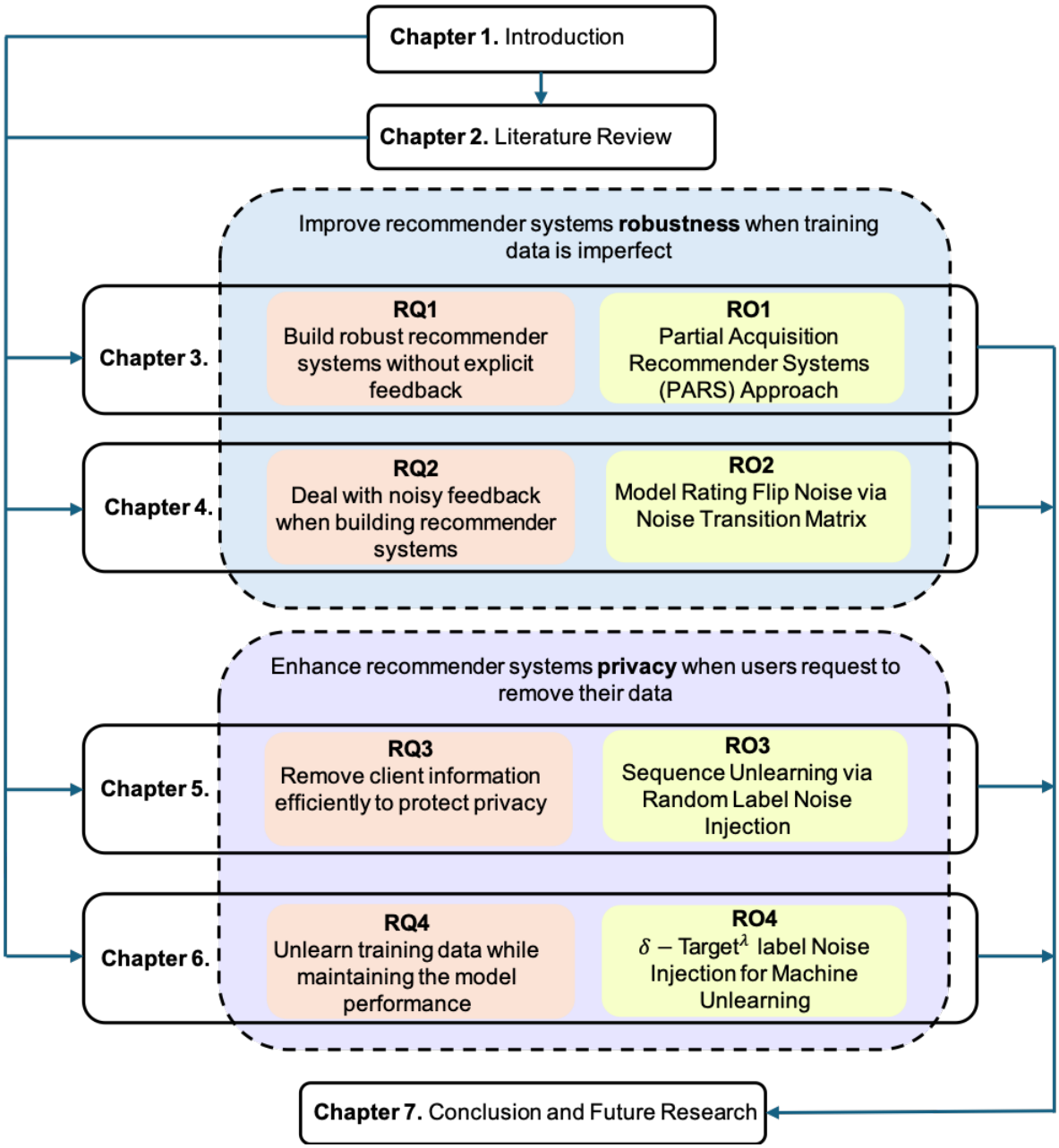

Figure 1.1: Thesis Structure.

CHAPTER 2

LITERATURE REVIEW

2.1 The Concept of Recommender Systems

With the progression of information technology and the expansion of the Internet, people have experienced the transition period from data sparsity to data overload (Zhang, 2013). Confronting excessive information, how to select relevant and preferred products or services effectively and efficiently is a big challenge for both individuals and businesses (Lu et al., 2020). In order to solve this problem, recommender systems have been introduced to help filter a great amount of information and pass the most relevant and useful data to target users. By way of analysing users' historical online behaviours, recommender systems are able to predict their fondness and interests so as to deliver personalised products or services (Resnick and Varian, 1997).

Recommender systems have initially been used in some E-commerce applications and nowadays they have been playing a more and more significant role in various areas (Schafer et al., 1999). Over the past 20 years, recommender systems have transformed the ways people conduct business, which catches extensive attention (Lu et al., 2015).

They have been applied widely in different areas such as commerce, government, library, social media, tourism industry, business services and so on. Companies like Amazon, eBay, YouTube, Netflix etc. have reaped tremendous benefits by employing robust and effective recommender systems (Lu et al., 2020).

In a nutshell, recommender systems are tools helping to connect users with the most relevant products or services. In the environment of information overload, recommender systems make predictions of users' preferences so as to generate a personalised recommendation list for them, which helps individuals or businesses to make better decisions and save a lot of time (Adomavicius and Tuzhilin, 2005). There are many recommendation algorithms available to work out users' top interesting lists. Based on different approaches and algorithms, recommendation systems can be roughly classified into four types: content-based, collaborative filtering, knowledge-based and hybrid techniques (Terveen et al., 2001). These four approaches will be elaborated on and discussed in the following sections.

2.2 The Framework of Recommender Systems

Recommender systems start with the step of obtaining information on users(U), items(I) and context(C). On the basis of these contents, recommender systems can be briefly viewed as a process: under certain contexts, regarding the huge amount of item information, the task is to build a model f (U, I, C) so as to make the predictions of users' preferences and generate Top N recommendation list (Wang, 2020). Therefore, it is natural to conclude that data sources(input), recommendation engine(model) and recommendation results(output) form the major parts of the recommender systems' framework (Lu et al., 2020).

2.2.1 Data Input

Data input is the collection of users, items and their interaction information, which shows in different forms and types. The recommender system will usually identify the features of users and items and create a profile of their information (Lu et al., 2020). For instance, there are several types of user features: demographic features include users' age, gender, race, etc., which are provided upon registration; user behaviour features involve their searching or buying history, their feedback on the products and so on (Xiang, 2012). Items are products or services put forward to users. The characteristics of items can be tangible or intangible. For example, tangible items can be movies, music, groceries, clothes, shoes and so on. On the other hand, intangible features can be knowledge, insurance, etc. (Lu et al., 2020).

When users interact with items, they will have either explicit feedback or implicit feedback (Oard et al., 1998). Explicit feedback pertains to the behaviour that users give a certain score to the products or services. Explicit feedback can only be given after the items have been purchased or consumed. Whereas implicit feedback can be provided without past buying behaviour. It refers to those activities, whether users search, click or not. If it is yes, then it shows positive implicit feedback. Otherwise, it's negative, which is usually discarded (Del Olmo and Gaudioso, 2008).

2.2.2 Recommendation Engine

The Recommendation engine plays a key role in recommender systems. To begin with, the information on users, items and context will go through a procedure of extraction, transformation, and loading, which are all collected and kept in the database (Jallouli et al., 2017). After that, the engine containing algorithms and models is to be trained, evaluated and deployed and online inferred (Wang, 2020).

The evaluations of the recommendation engine consist of offline evaluation and online

evaluation (Zangerle and Bauer, 2022):

(1) Offline evaluation: it is based on historically explicit or implicit feedback that was collected before. It requires no interactions between users and items. The algorithms and evaluation are conducted offline. Though it is simple to do the offline experiments several times so as to achieve evaluation objectives, the scope is relatively narrow, and the precision is comparatively low compared to the online evaluation.

(2) Online evaluation: Under the online evaluation context, users and items have real-time interaction. The information is collected and then used on a real-time basis to enhance the recommendation accuracy. Commonly, online evaluation is realised through A/B testing, which randomly chooses a limited number of individuals to be allocated to different recommendation system methods. The real-time interactions among the individuals are therefore recorded and examined. As a result, the feedback generated from different users is gathered and analysed to evaluate which recommendation approach is more persuasive.

2.2.3 Recommendation Output

After the process of recommendation input and engine, a recommendation list of users' favourite items (also known as the top-N recommendation list) is generated and offered to users to select the most relevant ones. In addition, another typical recommendation generation is presented as a type of product bundling that provides a combined package of items to users (Bai et al., 2019). The bundling recommendation list offers a quality range of products or services, enhancing customers' experience and ultimately boosting sales volume.

2.3 The Methodology of Recommender Systems

2.3.1 Content-based Recommender Systems

Content-based filtering technique is based on the users' previous buying behaviour or their recorded feedback, which is not affected by other users' information. By creating a profile to represent each user and item's features and characteristics, the content-based recommender (CBR) approach is able to identify items or products' similarities and therefore recommend similar new items to the users according to their previous preference records (Shu et al., 2018).

2.3.1.1 Application of the Content-based Recommender Systems

The content-based recommender system is applied widely in various sectors. The most typical example is recommending movies. The realisation of the CBR techniques in movie recommendations will be discussed briefly in this part. Firstly, it is essential to use some tags and features to describe items. For example, there are three movies that need to be put on tags: IronMan is under the tags of Action and Sci-fiction; Spider-Man is under the tags of Action and Sci-fiction; Kung Fu is under the tags of Action and Chow Sing chi. The second step is to calculate users' tag preferences. The user preference tag vector can be worked out based on users' previous behaviour of chosen movies. Finally, based on the items' feature tags and users' preference tags, the law of cosines can be used to work out the top-N recommendation list. A graphical representation of the movie recommendations is shown in Figure 2.1.

2.3.1.2 Evaluations of the Content-based Recommender Systems

There are several merits of the CBR method. First of all, it does not require the information of other individuals, working independently, which makes it easier to calculate the

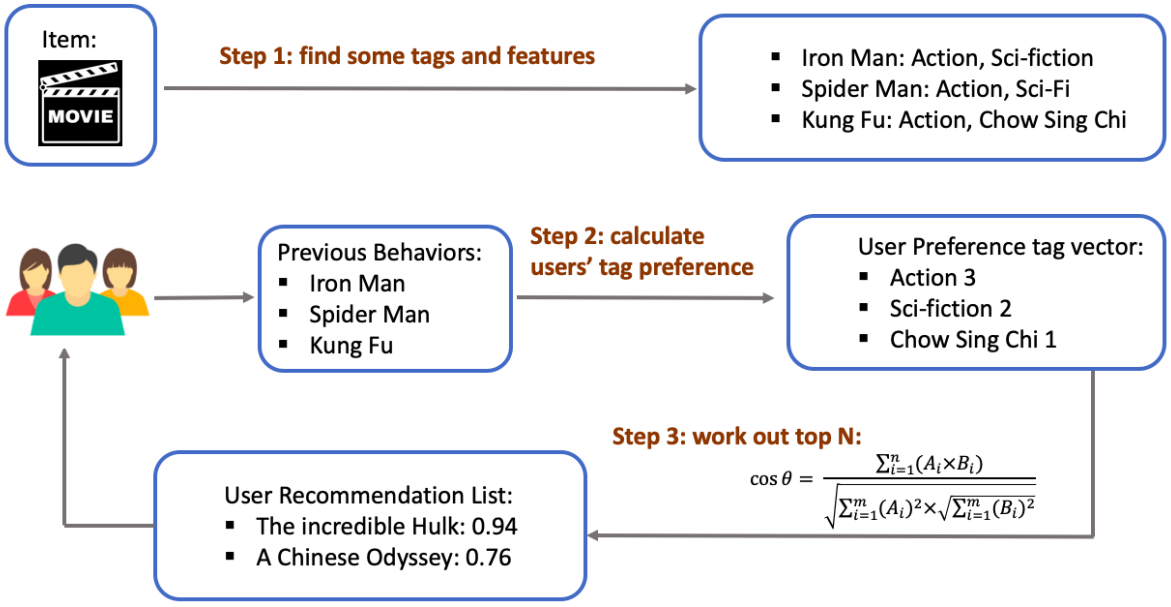

Figure 2.1: A graphical illustration of CBR application in movie recommendation

algorithms and thus faster to make the recommendations. Additionally, since the CBR is building profiles and creating tags of individual users, it identifies their preferences and interests better. Therefore, it's possible to make recommendations to those minority groups of users who have unique tastes. Finally, for those recently released and unpopular items, the CBR approach can match their similarity with users' tags and preferences and thus make appropriate recommendations.

However, CBR inevitably has some disadvantages. To begin with, the tags used to represent the items are sometimes hard to describe precisely. Especially for the implicit information, there are some challenges to obtain their tags (Shu et al., 2018). In some cases, it may need extra labour expense on the tag description. What's more, the CBR method is only focused on individuals' preferences, having no interactions with other individuals. Therefore, it has limitations in identifying and discovering users' potential interests (Lops et al., 2011). Lastly, for new users who do not have past behaviours, it is impossible to make recommendations for them.

2.3.2 Collaborative Filtering Recommendation Methods

To address some problems of the content-based approach, collaborative filtering methods have been introduced to provide more personalised and accurate recommendations. The collaborative filtering recommendation takes into consideration other users' feedback, ratings or opinions, thus filtering the huge amount of information so as to recommend the most interesting or relevant items to target users (Wang, 2020). There's an underlying assumption indicating that users who embrace similarity tend to buy similar products or services (Lu et al., 2020). Accordingly, the collaborative filtering recommendation (CF) relies on the past ratings or feedback of similar users so as to recommend relevant items to other users (Nilashi et al., 2013).

In terms of the difference in characteristics and algorithms, the collaborative filtering recommendation has been divided into two categories: memory-based collaborative filtering and model-based collaborative (Ekstrand et al., 2011), which will be discussed in detail in the following sections.

2.3.2.1 Memory-based CF

Memory-based CF is one approach that works out the likeness in users or in items based on the past ratings given by users; therefore, it can be classified into two forms: user-based CF and item-based CF (Lu et al., 2020). These procedures containing feedback, information about users and items are recorded and kept in memory (Nilashi et al., 2013).

The figure 2.2 is an instance illustrating the difference between user-based CF and Item-based CF. From the user side, Andy is the target user. In an online restaurant, Kate and Andy both bought noodles and sushi before and provided similar feedback. It shows that Andy and Kate are similar users who share similar tastes. Kate also likes hamburgers, which Andy didn't purchase before. Therefore, the hamburgers will

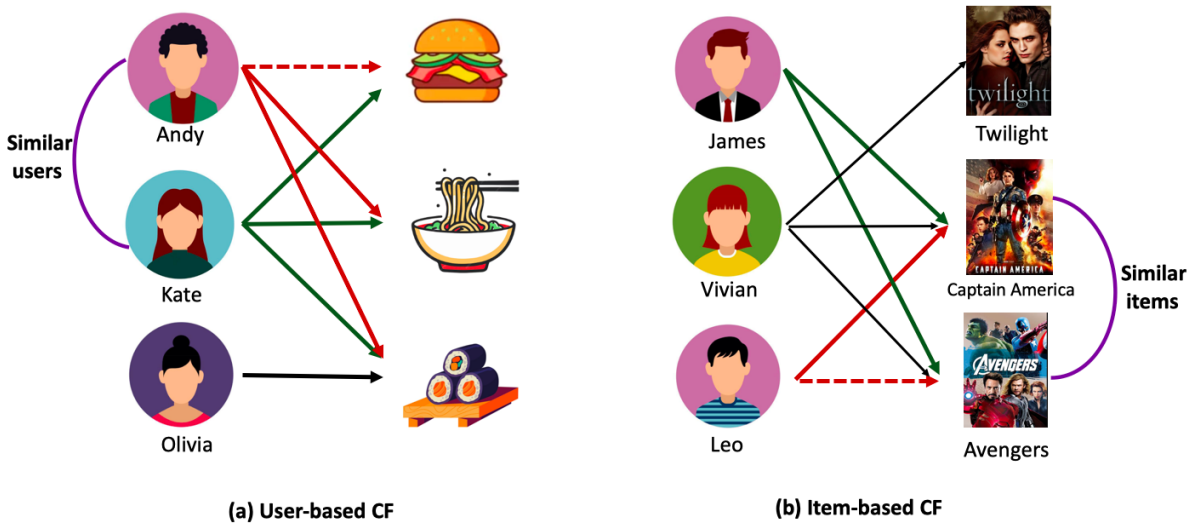

Figure 2.2: A diagram illustration of Memory-based CF

be recommended to Andy as he and Kate are recorded as similar users in the system. On the other hand, in the view of item-based CF, the movies "Captain America" and "Avengers" have shared similar ratings, and they are classified as similar items in the system. Therefore, for anyone who watched one of these two movies, the other one will be recommended to the user accordingly.

The above example shows the fact that the two types of CF methods (user-based and item- based) are assumed under the condition that similar users tend to consume similar products or similar items are attracted by like-minded users. The user-based CF is a process from user to user and then to the item (U2U2I). The procedure can be expressed as:

$$\text{sim}(u,s) = \sum_{u_i \in U} \text{sim}(u,u_i) \cdot \text{score}(u_i,s). \tag{2.1}$$

where u is the target user, u_i are other users and s represents the items.

For another, the Item-based CF is a process from user to the item and then to the item (U2I2I), which can be illustrated as:

$$\text{sim}(u,s) = \sum_{s_i \in S} \text{score}(u,s_i) \cdot \text{sim}(s_i,s). \tag{2.2}$$

2.3.2.1.1 User-based CF

The User-based method starts from the step of locating similar users who have similar ratings to the target user and then makes predictions of the target user's preference and creates a recommendation list (Nilashi et al., 2013). The process of how to work out the user-based CF approach is illustrated as follows:

(1) Step one: Compute user similarities.

The calculation of user similarities is the key part of collaborative filtering recommendation. Cosine similarity and Pearson correlation are the most common methods to work out user similarities.

Cosine similarity: As shown in Eq. 2.3, cosine similarity measures the similarity between users, who are represented by two vectors i and j. The distance between vectors i and j is the level of similarity between users. The angle of vectors i and j is smaller, the more similar the two users (Wang 2020).

$$\text{sim}(i,j) = \cos(i,j) = \frac{i \cdot j}{\|i\| \cdot \|j\|}. \tag{2.3}$$

The cosine similarity is easy to calculate. However, it fails to consider the mean factors and the differences in ratings given by users i and j.

Pearson Correlation: To address the limitation of cosine similarity, the Pearson correlation utilises the factors of mean and variance to improve the ratings' accuracy, to eliminate the skewing impact of users' ratings (Nilashi et al., 2013).

$$\text{sim}(i,j) = \frac{\sum_{p \in P}(R_{i,p} - \bar{R}_i)(R_{j,p} - \bar{R}_j)}{\sqrt{\sum_{p \in P}(R_{i,p} - \bar{R}_i)^2}\sqrt{\sum_{p \in P}(R_{j,p} - \bar{R}_j)^2}} \tag{2.4}$$

where $R_{i,p}$ represents the ratings given by user i and $\bar{R}_i$ is the average rating of all the items. P is the collection of all the items.

(2) Step two: Predict the rating

The method is based on the assumption that the target user has the same preference as like- minded users. Therefore, after the user similarity has been calculated, the rating prediction of the target user can be made based on the similar users' existing feedback. The most common way is to calculate the weighted average of the user similarities and similar users' ratings to get the target user's rating prediction (Wang 2020):

$$R_{u,p} = \frac{\sum_{s \in S}(W_{u,s} \cdot R_{s,p})}{\sum_{s \in S} W_{u,s}}, \tag{2.5}$$

where the weights $W_{u,s}$ is the similarity between the user u and user s; $R_{s,p}$ is the rating of the item p given by the user s.

(3) Step three: Generate the Top-N recommendation list.

After obtaining the target user's rating prediction, the recommendation list is generated by a ranking form based on the feedback on items that are new to target users (Lu et al., 2020).

The limitations of the user-based CF

(1) **Scalability**: Under the environment of the internet's application, the number of users is far more than the number of items. The user-based CF needs the computation of the matrix to maintain the same increasing pattern as the number of users and items (Sarwar et al., 2001). With the massive number of users and items, the expense to run and store the matrix of users' similarities is significantly high. Specifically, the growing number of users will probably lead to the storage space of the matrix increasing by n^2 (Wang 2020), which suffers critical scalability issues (Sarwar et al., 2001).

(2) **Sparsity**: In real-world applications, the vectors of users' historical information are more likely to be sparse. Especially for users with limited purchasing or clicking behaviours, the accuracy of locating similar users is very low (Sarwar et al., 2001), which makes user-based CF unsuitable for businesses requiring a significantly large amount of

feedback from users.

2.3.2.1.2 Item-based CF

Due to the technical limitations of user-based CF, item-based CF is more prevalent among companies. For example, companies like Amazon and Netflix are using item-based CF to realise their initial application phase of recommender systems. Unlike the user-based CF, the item-based CF is based on item similarities and then generates a top-N recommendation list.

The computation of item-based CF is a transposition of the user-based approach. The first step is to calculate the item similarities. The Pearson correlation is the most popular algorithm to work out the item similarities:

$$\text{itemSim}(v_i, v_j) = \frac{\sum_{u \in U_{v_i,v_j}} (R_{u,v_i} - \bar{R}_{v_i})(R_{u,v_j} - \bar{R}_{v_j})}{\sqrt{\sum_{u \in U_{v_i,v_j}} (R_{u,v_i} - \bar{R}_u)^2} \sqrt{\sum_{u \in U_{v_i,v_j}} (R_{u,v_j} - \bar{R}_{v_j})^2}}, \tag{2.6}$$

where U_{v_i,v_j} represents users give ratings to both items v_i and item v_j . The equation 2.6 also considers the average ratings of items v_i and item v_j, which is $\bar{R}_{v_i}$ and $\bar{R}_{v_j}$ respectively. The result of PCC ranges from -1 to 1. The closer to 1, the higher the similarity between these two items (Schafer et al., 2007).

After the item similarities are worked out, the next step is to predict the ratings of target user u on item v_i based on the nearest neighbours of item v_i. The weighted sum has been applied to ratings that correspond with the similarity s_{v_i,v_j} between items v_i and v_j (Sarwar et al., 2001). The equation is presented as follows:

$$P_{u,v_i} = \frac{\sum_{v_j \in V_u} (S_{v_i,v_j} \cdot R_{u,v_j})}{\sum_{v_j \in V_u} |S_{v_i,v_j}|}. \tag{2.7}$$

Once obtaining the rating prediction, the last step of generating the Top-N recommendation list is the same as the user-based CF.

The application of User-based CF and Item-based CF

Besides the difference in the techniques between user-based CF and item-based CF, their applications in practice are not similar as well (Wang 2020). On the one hand, as the user-based CF is making recommendations on the basis of users' similarities, it tends to have more social attributes, which users can understand better about the recently interesting stuff shared by the like-minded group. Though the new interesting topic is not favoured by these users before, the recommendation list will be refreshed according to similar users' recent updates. This kind of feature makes it quite suitable for news recommendation because users' interest towards news is usually dispersed. Compared to users' preferences and hobbies toward different news, the timeliness and headline of news are usually more important attributes. The user-based CF is more suitable for discovering and updating breaking news.

On the other hand, item-based CF is more feasible for applications that have stable interests and preferences. For instance, on Amazon's E-commerce website, it is effective and efficient to utilise item similarity algorithms to recommend relevant items to consumers who tend to search for certain types of products over a certain period. Another example is Netflix's recommendation for videos. Users' preferences and interest in watching movies tend to be stable. Therefore, it's more reasonable to use item-based CF instead of user-based CF to recommend similar styles of videos.

The application of item-based CF in Amazon

Facing the challenging online business environment of millions of consumers and products, Amazon has utilised item-based CF to provide personalised products and developed algorithms tailored to meet their needs to perform effective and efficient recommendations (Linden et al., 2003). As traditional recommendation methods have scalability problems in dealing with the enormous amount of user bases, Amazon im-

proved the item-based algorithms to be applied both online and offline. Amazon not only builds extensive computation in their systems offline, but it also has robust online components to address customers' needs in real-time. Therefore, they can have quicker reactions to handle the changes in consumer data and provide quality recommendations.

2.3.2.2 Model-based CF

While memory-based CF is focused on analysing similarities between users or between items, model-based CF is centred on building models to show the characteristics of both users and items (Koren et al., 2009). Model-based CF has been widely used due to its advantages of adequate scalability, high accuracy and more flexibility on a real-time basis (Koren et al., 2009). Two techniques will be discussed in the following parts: matrix factorisation on explicit feedback and Bayesian personalised ranking from implicit feedback.

2.3.2.2.1 Matrix factorization

The matrix factorisation has drawn tremendous attention from the public in the Netflix Prize contest in 2006 (Mehta and Rana, 2017). Computing on the foundation of the matrix, it brings in the concept of the latent semantic model (Hofmann, 2004), enhancing the ability to deal with the sparsity issues of the matrix and improve weak generalisation problems (Luo et al., 2015).

Koren et al. (2009) proposed a principal matrix factorisation model by locating users and items in a space that is represented by the latent vectors of both users and items. Therefore, it creates interdependence between users and items, which are defined as inner products in the space. If the distance between users and items is small, it shows their similarities are high, and thus the corresponding items will be recommended (Wang 2020). For the user matrix U and the item matrix V, the rating prediction given by user

u to item i is shown as follows:

$$\hat{r}_{ui} = q_i^T p_u, \tag{2.8}$$

where p_u is the corresponding row vector of user u in matrix U and q_i is the related column vector of item i in item matrix V. The equation 2.8 reflects the interrelationship between user u and item i. After this mapping has been fulfilled, the prediction ratings of any items can be made through this equation (Koren et al., 2009).

(1) **Singular Value Decomposition (SVD)**

Singular Value Decomposition is one of the approaches in matrix factorisation to identify the latent semantic interrelation between users and items (Cui et al., 2018). The association between users and items is used to make predictions of users' interesting items (Dixit and Jain, 2018), which expects the user-item matrix to be factored (Koren et al., 2009). In reality, the majority of users' pasting behaviour information is quite limited, making the user-item matrix shown to be sparse and thus the rating prediction based on that is lacking precision (Rodpysh et al., 2021). Moreover, the computation of the traditional SVD is very complicated, reaching a level of $O(mn^2)$. As a result, it's impractical for those e-commerce sites that own millions of users and items (Wang 2020).

(2) **Regularized SVD**

In order to solve the problems of traditional SVD, the regularised SVD that only focused on observed ratings has been proposed by Simon Funk in the Netflix campaign (Bennett et al., 2007). On the one hand, the regularised SVD has addressed the overfitting problem by incorporating regularisations to train the model to be more stable and regularised (Koren et al., 2009). The objective function is shown as follows:

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2). \tag{2.9}$$

To minimise the squared difference between the original rating r_{ui} and the inner product $q_i^T p_u$, it can lower the possibility of data distortions and missing values (Sarwar et al.,

2000). And λ is the parameter affecting the level of regularisation. The bigger the value of λ, the stronger the restrictions on the regularisation.

The stochastic gradient descent (SGD) can be utilised to optimise the equation 2.9 (Chen et al., 2012). The parameters with the learning rate are modified in an iterative way, moving oppositely against the gradient (Koren et al., 2009). The SGD is popularised because it saves running time and has simple computations when facing a large number of training samples, and it also enhances performance in terms of precision and convergence speed (Khan et al., 2019).

(3) **Improved Regularized SVD**

In the rating matrix of the recommender system, users tend to have different rating criteria. Different types of items are measured in inconsistent standards as well. For example, the rating difference between electronic products and daily necessities tends to be significant. These variances are known as biases which are not influenced by any interactions but affect the performance of the matrix factorisation (Paterek, 2007). In order to reduce these biases and reflect accurate values, the rating will be adjusted by considering the user bias and item bias. The equation is shown as follows:

$$r_{ui} = \mu + b_i + b_u + q_i^T p_u, \tag{2.10}$$

where μ is the overall average rating, parameters b_i and b_u are the item bias and user bias respectively. For example, the average rating μ of the movie "Avatar" is 3.8. "Avatar" is a top-rated movie with a tendency to be rated 0.6 (item bias b_i) higher than the average movie. But user Kate is a critical customer who tends to give a below-average rating; therefore, the user bias is 0.2. As a result, based on equation 3.10, the adjusted rating for "Avatar" is 4.2 (3.8+0.6-0.2).

(4) **Multi-level SVD**

Rodpysh et al. (2021) proposed that cold start and sparsity issues in CF can be solved by multi-level SVD. Cold start and sparsity are two underlying problems in

recommender systems (Kulkarni and Rodd, 2020). Cold start issues refer to the situation when new users or new items come into the system, it lacks sufficient data or information about their pasting behaviours, and thus it's difficult to make recommendations to them (Camacho and Alves-Souza, 2018). The sparsity problem occurs when there isn't enough feedback or ratings given by users to some items.

There are several approaches to address the cold start and sparsity issues. The first approach is incorporating the contextual information similarity evaluation with the users and items similarity measurement. The contextual information refers to the background information about activities, place, time and so on (Villegas et al., 2018), which helps to identify users' preferences in some circumstances and their motivations for some choices so as to mitigate the cold start and sparsity problems for new users and items (Raza and Ding, 2019). Additionally, the context feature SVD has been proposed to combine the features and attributes of users matrix, items matrix and contexts matrix to establish their similarity (Natarajan et al., 2020). Therefore, the matrices of users, items and contexts are employed as elements of a multi-level SVD matrix, together with the help of stochastic gradient descent to tackle cold start and sparsity problems (Rodpysh et al., 2021).

Evaluations of Matrix factorisation

Compared to memory-based CF, matrix factorisation has several advantages. Firstly, the generalisation ability is better, resolving the sparse data problem to some degree. Moreover, unlike memory-based CF, matrix factorisation does not require the storage space for the vast number of matrices of user similarity and item similarity, only needing to store the vectors of users and items. It reduces the space complexity from nm level to $(n+m)k$ level. Note that n and m represent the number of rows and columns of a matrix respectively, and k represents the dimension of the newly represented dat. Normally, $k << n$ and $k << m$. Lastly, matrix factorisation has better scalability and flexibility. The

output of matrix factorisation is vectors of users and items, making it easier to combine with other attributes or features.

However, similar to memory-based CF, matrix factorisation has the same challenges of incorporating users, items and context features to determine the similarities. Meanwhile, it also lacks sufficient historical data about users, making it difficult to generate effective recommendations. Nevertheless, as discussed in the previous section, Rodpysh et al. (Rodpysh et al., 2021) proposed a solution by employing multi-level SVD to diminish the cold start problems.

2.3.2.2.2 Bayesian personalised Ranking from implicit feedback

Most of the matrix factorisation approaches are dealing with explicit feedback that shows users' interests and preferences by ratings (Lerche and Jannach, 2014). In reality, however, explicit feedback tends to be sparse, especially for those cold-start users and items (He and McAuley, 2016b). For unobserved feedback, also known as implicit feedback, Bayesian personalised ranking (BPR) has been introduced to analyse users' preferences by a pair-wise ranking approach (Loni et al., 2016). Implicit feedback is captured spontaneously, like clicking, viewing, buying and so on, which only shows positive feedback and the negative one is omitted (Rendle et al., 2012). BPR plays a significant role in recommender systems to understand users' needs effectively. Nevertheless, there are many improvements that have been proposed by different authors so as to take full advantage of users' feedback.

Rendle et al. (2012) proposed that the BPR method can be optimised by maximising the posterior estimator. By utilising the likelihood function, a learning algorithm pertaining to BPR-OPT has been worked out. After that, the BPR-OPT has been optimised by stochastic gradient descent. Additionally, Krohn-Grimberghe et al. (2012) suggested

building multi- relational matrix factorisation to make BPR more adaptable. The proposal is mainly about broadening the BPR structure and framework to multiple relations so as to make it suitable and optimal for different assessment criteria. Similarly, Loni et al. (2016) extends the work of Krohn-Grimberghe et al. (2012) by proposing an idea to generate a BPR framework with Multi-Feedback. The initiative is that in a training process, to employ unary feedback making use of multi-channel so as to enhance the performance of BPR. The main contribution is that different types of evaluations have been identified by multiple channels, which are located at different levels of feedback.

2.3.3 Knowledge-based Recommender Systems

Knowledge-based recommendation techniques are based on the fields of knowledge about customers and products to provide personalised recommendations (Colombo-Mendoza et al., 2015). Unlike content-based and collaborative-based methods, the knowledge-based recommendation is independent of the historical information about users' buying behaviours and thus it addresses the cold-start and sparsity challenges to some degree (Bouraga et al., 2014). The products are usually fairly customised and not bought frequently. For example, when people want to buy houses, they have their own preferences and criteria about bedrooms, location, design, size and so on. It is unreasonable and irrelevant to make recommendations based on the previous ratings of the houses. Therefore, knowledge-based recommendation techniques are introduced for these products requiring specific field knowledge, like houses, cars, electronics, financial products and so on (Aggarwal, 2016). These items share similar characteristics that they are not purchased on a regular basis, their attributes or domain may be complicated, and they are usually highly customised (Lu et al., 2020).

Aggarwal (2016) proposed that knowledge-based recommendations can be classified into two types: constraint-based and case-based recommender systems. In constraint-

based recommender systems, different types of constraints or rules of products have been exerted so as to fulfil users' specific requirements (Felfernig and Burke, 2008). There are usually two types of constraints (Aggarwal, 2016). The first one is the constraints on item features. For example, users are looking for a house that was built after 2000 and specifically has a certain number of bedrooms. Another type of constraint is linking user characteristics to item features. For instance, younger investors tend to buy high-return and risky financial products.

For case-based recommender systems (Bridge et al., 2005), it relies on past problem-solving cases and uses similar experiences and solutions to tackle new problems (Lorenzi and Ricci, 2003). The process involves four steps: retrieve, reuse, adapt and retain (Lu et al., 2020). In the retrieval phase, when a new problem is presented, the system will search the databases and then assess previous cases to find the most similar one to the new problem specification (Burke, 2000). A range of product attributes, defined by customers, mainly formed the problem part of the case. Whereas, the case's solution part lies in the product (Bradley et al., 2000). The case-based recommendation technique is deemed as a form of content-based recommender system. Nevertheless, the case-based recommendation method is more personalised by focusing on the specific problem and solution attributes (Lorenzi and Ricci, 2003). Additionally, incorporating the ideas of domain knowledge and item similarity, it enhances users' interaction with products, having more flexibility and higher transparency (Smyth, 2007).

2.3.4 Hybrid Recommender Systems

Based on the previous discussion of different types of recommender system approaches, all methods have merits and limitations (Burke, 2002). For example, content-based techniques may not have cold-start issues for items, but they are not as accurate as collaborative filtering methods (Claypool et al., 1999). And collaborative filtering techniques

have higher precision in results, but they suffer from cold-start and sparsity problems (Gunawardana and Meek, 2009). Therefore, hybrid recommender systems are used to combine two or more recommender systems methods to reap greater benefits (Çano and Morisio, 2017). The most popular combination is linking the collaborative filtering approach with other recommendation methods to obtain complementary advantages (Lu et al., 2020).

2.4 Cross-domain Recommender Systems

Machine learning models are trained with data. By statistical learning theory (Vapnik, 2013), models will perform better with more training data. Although the overall data volume is significantly bigger than before, in the era of big data (Manyika et al., 2011), data is still limited for many real-world applications. For example, when start-ups build recommender systems, data size will be small at the beginning because of limited users. Transfer learning (Pan and Yang, 2010) aims to improve the performance of the target domain task, which has limited data information, by transferring knowledge from a different but related source domain that has sufficient data information. Cross-domain recommender systems make use of the advantage of the philosophy of transfer learning. They, therefore, have been widely employed for real-world applications with limited data. In this section, we will quickly review the knowledge of transfer learning and cross-domain recommender systems algorithms.

2.4.1 Transfer Learning

In this subsection, we will survey transfer learning from the following perspectives: transfer learning settings, domain shift scenarios, and the methods to extract the invariant information across domains. Research in transfer learning has expanded rapidly, resulting in a vast array of scholarly work. We cannot provide a comprehensive survey

due to the space limit and the scope of the thesis. More details can be found in the surveys (Pan and Yang, 2010; Zhuang et al., 2020).

2.4.1.1 Transfer learning settings

Transfer learning is motivated by humans' ability to transfer experience (or knowledge) from one task to another one. In artificial intelligence, transfer learning is designed to transfer data knowledge from the source domain to the target domain. The target domain is defined as the data domain of the targeted task that we would like to address. The source domain refers to the data domain, which is different from the target domain but can be used to boost the performance of the target task. The source domain and target domain are different but closely related (which will be carefully reviewed in Section 2.4.1.2). More specifically, the source domain usually has sufficient data, while the target domain has limited data. By only exploiting the target domain data, we cannot train models with good performance for the target task. To do so, we need to make use of the source domain data.

Many different settings of transfer learning have been studied in the literature. We intuitively introduce some representatives. The most accepted setting for transfer learning is that there is limited labelled training data in the target domain and sufficient labelled training data in the source domain (Torrey and Shavlik, 2010). When the target domain only has instances but no labels, the setting is called unsupervised transfer learning (Ganin and Lempitsky, 2015). When sourcing domain data is not available, but a model trained by the source domain data is available for further exploitation for the target task, the setting is called hypothesis transfer learning (Du et al., 2017). Super models (Hinterstoisser et al., 2018), which have achieved many successes, can perform well in downstream tasks with fine-tuning. This is a typical example of the deployment of hypothesis transfer learning. If the source domain data and models cannot be directly accessed to train the target model, the privacy of the source domain data can be largely

preserved. This setting is called privacy-preserving transfer learning (Dong et al., 2021). There are some learning settings that are closely related to transfer learning, for example, multi-task learning (Ando et al., 2005), learning to learn (Baxter, 2000), and life-long learning (Pentina and Lampert, 2014).

2.4.1.2 Domain shift scenarios

To ensure the success of transfer learning algorithms, we should analyse the changing and unchanging components across the source domain and the target domain. The unchanging component can be directly transferred from the source domain to the target domain, while the changing component cannot. We need to extract the invariant information contained in the changing component, which will be discussed in the next subsection. Let P^S_{XY} and P^T_{XY} be the joint data distributions of the source domain and target domain, respectively, where X represents the variable for instances, Y is the variable for labels, the superscript S denotes the source domain, and superscript T denotes the target domain. The joint distribution P_{XY} can be decomposed into $P_{Y|X}P_X$ and $P_{X|Y}P_Y$ by the conditional rule of probability. Generally, the conditional distribution $P_{Y|X}$ (or $P_{X|Y}$) is entangled with the marginal distribution P_X (or P_Y), which means when one of the distributions changes the other one will also change. Thanks to the modularity of causal systems (Schölkopf et al., 2012), they could be disentangled under certain causal structures. Specifically, when X is the cause of Y, $P_{Y|X}$ and P_X will be disentangled, which means that changing one of the distributions will not affect another one. When Y is the cause of X, $P_{X|Y}$ and P_Y will be disentangled.

If the class posterior probability $P_{(Y|X)}$ remains unchanged across domains while the marginal probability P_X changes, i.e., $P^S_{(Y|X)} = P^T_{(Y|X)}$ and $P^S_X \neq P^T_X$, the scenario is called covariate shift (Gretton et al., 2009). If the class posterior probability $P_{(Y|X)}$ changes across domains while the marginal probability P_X remains unchanged, i.e., $P^S_{(Y|X)} \neq P^T_{(Y|X)}$ and $P^S_X = P^T_X$, the scenario is called model shift (Wang and Schneider,

2014). If the class conditional probability $P_{(X|Y)}$ remains unchanged across domains while the class prior probability P_Y changes, i.e., $P^S_{(X|Y)} = P^T_{(X|Y)}$ and $P^S_Y \neq P^T_Y$, the scenario is called target shift (Zhang et al., 2013). If the class conditional probability $P_{(X|Y)}$ changes across domains while the class prior probability P_Y remains unchanged, i.e., $P^S_{(X|Y)} \neq P^T_{(X|Y)}$ and $P^S_Y = P^T_Y$, the scenario is called conditional shift (Gong et al., 2016). When we know the components that change across domains, e.g., via prior knowledge or domain expertise, we can design effective transfer learning algorithms by correcting the changes.

Although many transfer learning algorithms are designed without implicitly modelling the changing and unchanging components, they have close relationships with the aforementioned scenarios or their variants. For example, self-taught learning (Raina et al., 2007), which extracts basic vision structures from the source domain and makes use of them for the target domain, could be treated as a special case of model shift. The widely used pre-training technique (Erhan et al., 2010) can be explained by the fact that the class posterior probabilities $P_{(Y|X)}$ do not change significantly across domains.

2.4.2 Methods Extracting Invariant Information

To make use of the changing component, researchers propose to extract invariant features (Muandet et al., 2013) or correct the changing component to be the same (Li et al., 2018). We will introduce their philosophies in the following.

To extract invariant features, the kernel mean matching (Gretton et al., 2009) method has been proposed, which has a theoretical guarantee. The basic idea is that there is a bijection between the distributions and their expectations in the universal kernel Hilbert space (Gretton et al., 2009). Intuitively, if two distributions have the same expectation, the distributions are the same. When we extract features to have the same expectation in the universal kernel Hilbert space, we can ensure that the extracted features will have

the same distributions across domains. The mechanism of the generative adversarial network (GAN) (Goodfellow et al., 2020) has also been used for extracting invariant features. In GAN, there is a discriminator to distinguish fake images vs real images. There is a generator that tries to generate fake images that are similar to the real ones and fool the discriminative network. If we replace the generator with an extractor to extract features from the source and target domains, by the philosophy of GAN, invariant features across domains can be extracted when a well-trained discriminator cannot distinguish the extracted features (Bousmalis et al., 2017).

To correct the changing component to be the same. An important reweight method has been proposed (Zhang, 2013). If a distribution P' of the source domain is different from the corresponding distribution P of the target domain, we can use the importance reweight method to add a weight P/P' to the source domain distribution such that the reweighted source domain distribution $P/P' \times P'$ will become equal to the target domain distribution P. The key part is to learn the weight P/P'. Given data from the source and target domains, the weights can be learned by kernel mean matching with theoretical guarantees (Zhang, 2013). The weights can also be learned from the meta-learning method (Ren et al., 2018).

2.4.3 Cross-domain Recommender System Algorithms

The philosophy of transfer learning can help solve two major issues for recommender systems. The first one is that the target task has limited data, which is known as the data sparsity or cold-start problem (Jafri et al., 2022). The second one is that we could transfer knowledge from related tasks to further boost the performance of recommender systems in multiple different but related domains (Zhang, 2018).

However, the transfer learning methods developed in the transfer learning community cannot be directly employed for cross-domain recommender systems. The reason is

that there is usually no feature available in the target domain other than ratings or preference (Lu et al., 2015). To address the issue, many efficient and effective cross-domain recommender systems have been designed in the community of recommender systems. We would like to divide the cross-domain recommender systems algorithms into two categories: (1) cross-domain recommender systems with direct feature matching, i.e., via side information features; (2) cross-domain recommender systems without direct feature matching.

Although there is usually no feature available in the traditional setting of recommender systems, we could make use of side information, which contains features. For example, we could use item attributes if available (Singh and Gordon, 2008), collect extra user information from the social network (Jiang et al., 2015), or item information by exploiting the user-generated tags (Abel et al., 2013). Given the side information, data information can be transferred across domains. Specifically, given a user-item rating matrix and an item-attribute matrix, the cross-domain recommender systems can be designed by factoring the matrices in the constraint that they share the same item parameters because they share the same items. In other words, if we have sufficient items, then their attribute and good item parameters can be learned. Using it to help factorise the user-item matrix will result in better user and item parameters (Singh and Gordon, 2008). The same idea can be used to exploit the side information from the social network and the item information by exploiting the user-generated tags.

If there is no side information available for cross-domain recommender systems, methods are designed to extract the transferable user or item patterns to transfer knowledge across domains. For example, using the idea of dictionary learning (Kreutz-Delgado et al., 2003), the method of code book transfer has been proposed for cross-domain recommender systems by clustering users and items into groups and extracting common bases as group-level knowledge (Li et al., 2009). Deep transfer learning has been

employed to design cross-domain recommendation systems by sharing some common parameters across domains to address the data sparsity and cold-start issues (Jafri et al., 2022). If there are overlaps of users or items across the domain, data knowledge can be transferred by matching the users or items across domains (Li and Lin, 2014). The rationale is that the matching will regularise the target domain learning to ensure a good generalisation (Mohri et al., 2018).

2.5 Robust Recommender Systems with Noisy Data

The last decade has witnessed the successful development of building recommendation systems from reliable data. However, in the era of big data, there are plenty of data with imperfect information, e.g., noisy attributes, inaccurate ratings, distribution/domain shift, malicious attacks, and so on. For example, users may randomly click ratings without carefully showing their reflections, which will introduce the noisy rating problem; users' tastes may change quickly for short videos in TikTok and Kuaishou, which may introduce the domain shift problem. Those imperfect but large-volume data contain invaluable information to explore for building recommender systems. In other words, the value of big data cannot be fully maximised by just ignoring this data. We propose to exploit those imperfect data to build robust recommender systems. In this subsection, we will briefly survey how to deal with noisy data and then the robust recommender systems algorithms.

2.5.1 Dealing with Noisy Data

There are two basic intuitions to deal with noisy data. The first one is to cleanse the noisy data/extract confident data (which is likely to be clean). The second one is to design robust learning algorithms that are robust to the noise. It is usually challenging to cleanse data or extract confident data, as we do not know which data is clean or which

is contaminated. To do so, we need to know the properties of clean data. For example, providing a small set of clean data or knowing that complex models (e.g., deep models) will fit the majority of data first (we often assume that the majority of data is clean) (Han et al., 2018). These methods have the drawback that we cannot make use of the valuable information contained in the contaminated data.

To design robust learning algorithms, the straightforward ways are to employ regularizers (Golub et al., 1999) or early stopping methods (Xia et al., 2021) to prevent the models to overfit the noise. Robust loss functions will also be helpful to reduce the side effect of noise, e.g., the Huber loss (Barron, 2019). Robust loss functions will affect the gradient information during the optimisation procedure (Guan et al., 2017). To better handle the complex neural networks and make them robust, interventions have been designed directly on the gradient information, e.g., by clipping the gradient (Menon et al., 2020). It is also intuitive to assign clean data with more weight (Collins et al., 2020). When we know there are specific patterns of the noise, we can also model the noise and reduce its effect with theoretical guarantees (Murphy, 2012).

We would like to discuss a specific noise here, which is called adversarial noise (Madry et al., 2017). It is demonstrated that adversarial noise is imperceptible to human eyes but can be used to influence human decision-making by exploiting vulnerabilities in an individual's habits and patterns (Dezfouli et al., 2020). Although the possibility of designing user rating profiles to maliciously manipulate the output of recommender systems was studied two decades ago (Hurley, 2011), there are significant development of adversarial noise in recent days. The noise, therefore, should be carefully handled when we design robust recommender systems. Many techniques in the adversarial learning community (Chakraborty et al., 2018) can be borrowed directly from the community of recommender systems.

2.5.2 Robust Recommender Systems

In this subsection, we quickly survey the robust recommender systems algorithms. Many of the methods developed for dealing with noisy data can be directly used for building model-based robust recommender systems (Burke et al., 2015), e.g., the regularizers, early stopping methods, and robust loss functions. However, many of them cannot be used for building robust recommender systems because there is usually no feature information but preferences or ratings in the setting of recommender systems, e.g., the adversarially robust methods, because the adversarial noise is often additive to features. We briefly introduce three categories of robust recommender systems algorithms: (1) rating fraud detection; (2) robust recommender systems algorithms; (3) adversarially robust recommender systems, which correspond to the aforementioned three kinds of methods for dealing with noisy data.

The rating fraud detection methods can be roughly divided into two categories, i.e., behaviour-based fraudster detection methods and network-based fraudster detection methods. The former methods explicitly utilise the behaviour information to detect rating fraud, e.g., rating behaviour patterns will be exploited (Rayana and Akoglu, 2015). More details can be found in the review (Jiang et al., 2016). The latter methods implicitly model the property of rating fraud by exploiting neural networks, e.g., ranking the reliability of ratings (Kumar et al., 2018). More details can be found in the survey (Akoglu et al., 2015).

Many robust recommender systems algorithms are built upon the techniques developed in the robust machine learning community, for example, by using robust loss functions (Li et al., 2022b). The M-estimator has been employed to design robust matrix factorisation methods (Mehta et al., 2007). Early stopping and regularizers are also designed to prevent deep model-based recommender systems from overfitting noise data (Rendle and Schmidt-Thieme, 2008). A causal mechanism has been used to design

causally responsible recommender systems (Xiao and Wang, 2022), which are robust to noise.

Adversarial attacks are imperceptible to the human eye and would result in the maliciously targeted output of recommender systems. Recently, it is a hot research topic of how to build adversarially robust recommender systems. The research in this direction can be divided into two streams. The first one is to design different attack methods to fool existing recommender systems (Di Noia et al., 2020). The aim of the attack is to encourage and inspire more robust recommender systems. The other stream is to defend against the attacks. One of the most effective methods to boost adverbial robustness is called adversarial training (Bai et al., 2021a). Many different variants of adversarial training methods have been specifically designed for building adversarially robust recommender systems (Wu et al., 2021a; Tang et al., 2019).

CHAPTER 3

PARS: PARTIAL-LABEL-LEARNING-INSPIRED RECOMMENDER SYSTEMS

Recommender systems are widely required and deployed to address real-world problems. In this Chapter, we study a new yet challenging real-world setting for recommender systems, where only user browsing histories are available without any explicit feedback. No item acquisition information, e.g., purchasing or rating, is given. By assuming that user browsing sequences are likely to contain the items to acquire, we draw an analogy to the setting of partial label learning in weakly supervised learning. This enables us to train reliable recommender systems only using browsing histories. We term the proposed method as Partial Acquisition Recommender System (PARS). Empirical results on real-world benchmark datasets show the effectiveness of the proposed method. Surprisingly, we also show that the proposed method even surpasses some baselines using item acquisition information.

3.1 Introduction

Recommender systems (RS) have become essential to our modern lives, significantly reshaping how individuals acquire products, services, and content. Fundamentally, these systems act as information filtering tools, designed to provide personalised products or services, helping to address the growing problem of information overload (Resnick and Varian, 1997). RS have various applications (Lu et al., 2015), e.g., from prominent digital commerce such as Amazon and Alibaba to powerful media streaming platforms like Netflix and YouTube (Roy and Dutta, 2022). Their pervasive impact highlights their substantial market value and critical role in improving user engagement, increasing sales, and enhancing satisfaction, establishing them as a foundational element of modern e-commerce and digital content delivery.

To achieve satisfactory application performances, RS need to capture user interests precisely (Claypool et al., 2001; Lu et al., 2024, 2025). Traditionally, RS rely heavily on explicit acquisition feedback (Liang et al., 2021), such as preferences, ratings, purchasing information, etc., to build effective recommendation models. However, in real-world settings, due to data accessibility (Huang et al., 2024c) or privacy factors (Dwork, 2006), there are plenty of cases where explicit acquisition feedback is unavailable (Núñez-Valdez et al., 2018). For example, users may browse product pages, read articles, or watch video clips online without explicitly indicating a preference, leaving a rating, or making a purchase. For these situations where only browsing histories are available without any explicit acquisition information, it is demanding and important to build effective RS.

Although the aim of building RS based solely on browsing histories is compelling and appealing, it is infeasible to achieve it without any assumptions or constraints. For example, if users randomly browse products that they are not interested in, it is impossible to learn any useful item acquisition information from such browsing data. To tackle this issue, we make a crucial and realistic assumption: users are more likely

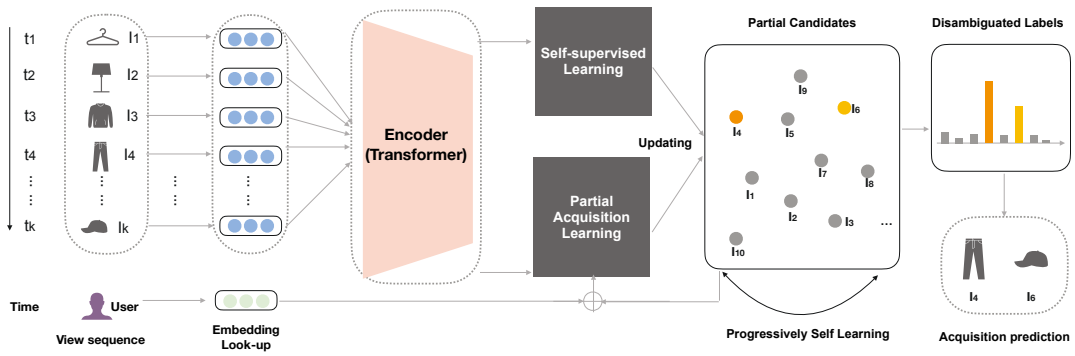

Figure 3.1: Illustration of the proposed PARS framework. The self-supervised learning part continuously updates the item embeddings. The partial label learning part is used to update the pseudo conversion probabilities gradually.

to browse products that interest them, i.e., the majority of browsing histories contain items that the users would like to acquire. We will discuss the assumption later to show it is weak. If this hidden item acquisition information can be effectively extracted from the browsing histories, it becomes feasible to build an accurate and robust RS without relying on explicit feedback.

To understand why the latent item acquisition information can be extracted, we draw an analogy of the proposed setting to a weakly supervised learning setting termed as partial label learning (Xu et al., 2019). In partial label learning, each instance has been assigned a set consisting of multiple labels while the set contains the true label. The goal is to learn a classifier that can predict true labels and generalise to unseen instances. It has been proven that effective learning is feasible for this setting (Cour et al., 2011). Later studies have extended the setting where the majority of label sets contain the true labels (Xu et al., 2023a). It is intuitive that the weak supervision information, i.e., the majority of label sets contain true labels, can be exploited to build a reliable classifier. In our setting, we have an analogy that the majority of user browsing histories contain the items to acquire, making it feasible to extract item acquisition information to build a reliable RS.

In this Chapter, we propose Partial Acquisition Recommender Systems (PARS), a

novel framework that, for the first time, enables reliable RS only exploiting browsing histories. The key innovation of PARS lies in its ability to learn conversion probabilities (or item purchase probabilities) from browsing to transaction without directly giving supervision of item acquisition. As shown in Figure 3.1, PARS introduces a partial label learning module combined with a self-learning mechanism that iteratively refines conversion probability estimates for items in user browsing sequences. Specifically, a masked self-learning part is used in the self-supervised learning part to capture temporal dynamics and contextual patterns in user browsing sequences. The partial label learning part ensures that initial noisy estimates are gradually refined as the model exploits the weakly item acquisition information that the majority of user browsing sequences contain the items to acquire. This approach represents a paradigm shift from supervised to unsupervised RS, making PARS particularly suitable for real-world scenarios where conversion feedback is unavailable.

Empirical results on popular real-world benchmark datasets consistently show the effectiveness of PARS. Since PARS is the first methodology to enable building RS only using browsing histories, there is no straightforward baseline. A related topic is conversion rate (CVR) prediction (Wen et al., 2019), which aims to estimate the probability that a user will complete a purchase after being exposed to an item. However, CVR needs the supervision of direct item acquisition labels in the training data. We show that PARS, without using any direct item acquisition information, will outperform CVR methods which use item acquisition information during the training phase.

Our work also has interesting implications. Firstly, it tackles a specific form of cold-start problem, where RS struggle to make predictions for new users with no acquisition data but do have early browsing histories (Rajendran and Sundarraj, 2021). By utilizing only browsing histories, our method can produce meaningful recommendations. Secondly, our approach introduces a novel privacy-preserving mechanism. Instead of collecting

and storing sensitive item acquisition data such as purchase records or explicit ratings, it relies solely on user browsing histories, which are generally less privacy-intrusive and can still yield strong recommendation performance. As a result, this capability fosters greater user trust and expands their applicability to a wider range of real-world contexts where explicit item purchase information is either limited by policy or considered ethically inappropriate (Ye et al., 2025).

In summary, our main contributions are as follows: (1) we formalise the problem of building RS using only browsing histories; (2) we propose a principled framework to recover acquisition information under realistic assumptions; and (3) we empirically validate our approach through comprehensive experiments, showing its robustness and real-world applicability.

3.2 Background

In this section, we will briefly introduce the related work beyond the traditional recommendation models, i.e., conversion rate prediction, self-supervised learning for sequential recommendation (SR), and partial label learning.

3.2.0.1 Conversion Rate Prediction

Conversion rate (CVR) prediction (Wen et al., 2019) aims to estimate the probability that a user will complete a purchase after being exposed to an item. CVR faces two fundamental challenges: sample selection bias, e.g., (Ma et al., 2018b,a; Wen et al., 2020; Wang et al., 2022a; Su et al., 2024; Huang et al., 2024a), and data sparsity, e.g., (Guo et al., 2017; Zhu et al., 2023; Ouyang et al., 2023), as conversion labels are only observed for purchased items.

Despite their success, all existing CVR methods share a fundamental requirement: they require explicit conversion labels during training. This requirement is frequently

violated in practice due to delayed conversions, cross-device purchases, and privacy constraints etc. Our work challenges this requirement by proposing a method that can learn purchase patterns without any explicit conversion labels.

3.2.0.2 Self-Supervised Learning for Sequential Recommender Systems

Our study involves user browsing sequences (Ye and Lu, 2023). Therefore it is closely related to sequential RS (Wang et al., 2019), which aim to predict users' future interactions based on their historical behaviour sequences. Recent advances have shown that self-supervised learning (Gui et al., 2024) can significantly improve representation learning without requiring explicit item acquisition labels.

Inspired by masked language modelling in NLP, masked sequence modelling has been proposed to SR (Kang and McAuley, 2018; Sun et al., 2019a). Another line of work employs contrastive learning to enhance sequence representations (Xie et al., 2022; Liu et al., 2021).

Since our setting does not have any access to explicit item acquisition information, we employ masked sequence modelling to help learn robust item embeddings.

3.2.0.3 Partial Label Learning

Partial label learning (PLL) (Xu et al., 2019) is a specific weakly supervised classification problem (Zhou, 2018; Ye and Lu, 2024). In PLL, each training instance is associated with a set of candidate labels, one of which is correct. This paradigm naturally handles label ambiguity in real-world applications.

Early PLL approaches can be categorised into two strategies: averaging-based methods (Cour et al., 2011) that treat all candidate labels equally, and disambiguation-based methods that iteratively refine label confidences (Zhang et al., 2016). Theoretical analyses have been provided to show the learnability, consistency, and generalisation of PLL algorithms (Cour et al., 2011; Lv et al., 2020; Feng et al., 2020; Xu et al., 2021a).

Existing PLL methods are primarily designed for multi-class classification with balanced datasets. We are the first to introduce it to the community of RS.

3.3 Partial Acquisition Recommender System (PARS)

In this section, we first setup the research problem. Then, we discuss the assumption enabling the idea of partial label learning (PLL) to extract item acquisition information. To learn user and item embeddings to estimate item acquisition probabilities, we design a transformer-based framework and employ masked language modelling and PLL to help learn semantic embeddings in an unsupervised manner.

3.3.1 Problem Formulation

Let $U = \{u_1, u_2, ..., u_n\}$ denote the set of n users, and $I = \{i_1, i_2, ..., i_m\}$ denote the set of m items. Each user $u \in U$ is associated with a session consisting of a sequence of interacted items $S_u = \left\{i_1^u, i_2^u, ..., i_{|S_u|}^u\right\}$, where $i_t^u \in I$ represents the t-th item in user u's interaction sequence. Let $L = \max\left\{|S_u| \,\middle|\, u \in U\right\}$ representing the maximum sequence length. We padd all the sequence to the length of L, i.e., $S_u = \left\{i_1^u, i_2^u, ..., i_L^u\right\}, \forall u \in U$.

We aim to learn which items users are likely to purchase based only on the observed interaction sequences $\{S_u | u \in U\}$. This setting is quite realistic in many e-commerce scenarios where purchase signals are delayed, missing, or privacy-protected. Given a user u with interaction sequence S_u, our goal is to predict an acquisition vector $\hat{\mathbf{p}}_u \in \mathbb{R}^L$ where $\hat{\mathbf{p}}_{u,i}$ represents the probability that user u will purchase item $i, i \in \{1, 2, \ldots, L\}$. The learning problem can be formulated as

$$f_\theta : S_u \mapsto \hat{\mathbf{p}}_u \tag{3.1}$$

where θ represents the parameters of model f. We term the problem as Partial Acquisition Recommender Systems (PARS) learning.

3.3.2 An Analogy to Partial Label Learning

We introduce the setting of PLL and then draw an analogy of PARS to PLL.

We misuse notation slightly to set up PLL for easy comparison. Specifically, let X be the instance space and Y the label set of multiple classes. Given training data $\{(\mathbf{x}_i, S_i) | 1 \leq i \leq n\}$, where $\mathbf{x}_i$ represents an instance, S_i a set of L possible labels, n the size of instances, PLL aims to learn a multi-class classifier $f : X \to Y$. It has been proven that if the majority of the label set S contains true labels, a reliable classifier f can be learned (Xu et al., 2023a).

We can see that PARS and PLL have very similar settings, where users in PARS correspond to instances in PLL; an item sequence that contains items the user may purchase corresponds to a label set that contains the possible labels for an instance; both settings have to exploit the underlying true acquisition/label information from the interaction sequence/label set. However, in PLL, it is required that the majority of label sets should at least contain one true label; while in PARS, it is not clear if the majority of sequences will contain at least one item purchased or not, because PARS studies real-world applications where data distribution cannot be controlled.

If **we assume that the majority of sequences contain at least one item purchased**, we can adapt the methodologies designed for PLL for PARS. It looks like the assumption is strong. However, in the next subsection, we will show that the popular benchmark datasets in RS used in the Chapter implicitly satisfy the assumption.

3.3.3 The Assumption is Weak

In this Chapter, we study the three popular datasets in RS, i.e., YooChoose[1], RetailRocket[2], and Taobao[3].

[1] https://www.kaggle.com/chadgostopp/recsys-challenge-2015
[2] https://www.kaggle.com/retailrocket/ecommerce-dataset
[3] https://tianchi.aliyun.com/dataset/649?lang=en-us

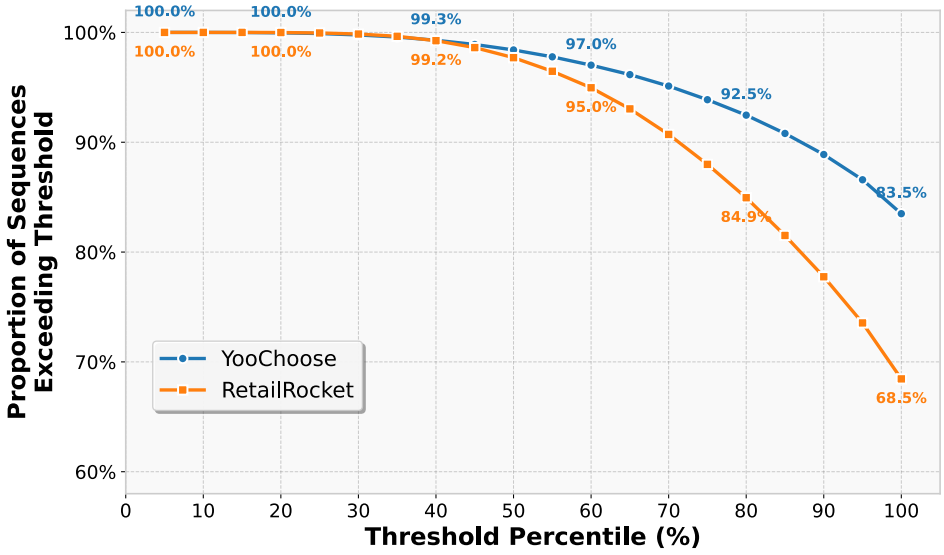

Figure 3.2: The vertical axis shows the percentage of sequences that contain at least one either positive or pseudo-positive acquisition label. The horizontal axis represents different ways to calculate the threshold for pseudo-positive acquisition labels. $x\%$ means the threshold equals the average value of the smallest $x\%$ of the predicted scores of the items with true positive acquisition labels. Note 100% means the threshold equals the average value of all the predicted scores of the items with true positive acquisition labels.

These datasets represent diverse user behaviours and purchase patterns across different platforms and regions. More details can be found in the experimental part.

All three datasets have the ground-truth item acquisition labels. Note that for PARS, we do not use the labels to train the model. We found that Taobao has 75.45% of the interaction sequences containing at least one item marked as purchased. However, YooChoose and RetailRocket have less than 50%, i.e., YooChoose has 20.98% and RetailRocket has 5.57%. It seems the assumption does not hold that the majority of sequences contain at least one item purchased. However, we argue that in real-world scenarios, the browsed item sequences are likely to contain the item the user would like to purchase, even though the transactions have not been made, e.g., the users may purchase the items on other platforms or at a later time.

To verify the above argument, we have checked the predicted item acquisition probabilities $\hat{\mathbf{p}}_u$ for the training data and found that many items have predicted acquisition

probabilities higher than those with positive ground-truth item acquisition labels. We can assign pseudo-positive acquisition labels to the items. As shown in Figure 3.2, we can see that the vast majority of items have at least one either positive or pseudo-positive acquisition label, making our above assumption implicitly hold true. Therefore, the methodologies designed for PLL can be borrowed to address PARS. In the following, we will show how to learn user and item embeddings and how to use the PLL philosophy for PARS.

3.3.4 Transformer-based User Embedding

To obtain user embeddings capturing item and sequential information, we use the following item and position embedding

$$\mathbf{e}_t = \mathbf{Z}_{\text{item}}(i_t^u) + \mathbf{Z}_{\text{pos}}(t) \tag{3.2}$$

where $\mathbf{Z}_{\text{item}} \in \mathbb{R}^{d \times m}$ is the item embedding matrix (with an additional token embedding for masking), $\mathbf{Z}_{\text{pos}} \in \mathbb{R}^{d \times L}$ is the positional embedding matrix, and d is the dimension of the embedding, $\mathbf{Z}(x)$ represents the x-th column of $\mathbf{Z}$.

We further employ a multi-layer Transformer encoder to enhance the expressive power of the embeddings, i.e.,

$$\mathbf{H}^{(l)} = \text{TransformerLayer}^{(l)}(\mathbf{H}^{(l-1)}), \quad l = 1, ..., N \tag{3.3}$$

where $\mathbf{H}^{(0)} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_L]$, N is the number of Transformer layers, and each layer consists of multi-head self-attention and feed-forward networks with residual connections and layer normalisation.

To obtain a global representation of the user's sequence, we employ the average pooling over the non-padding part of $\mathbf{H}^{(N)}$, i.e.,

$$\mathbf{g}_u = \frac{1}{\sum_{t=1}^{L}(1-m_t)} \sum_{t=1}^{L} (1-m_t) \cdot \mathbf{H}_t^{(N)} \tag{3.4}$$

where $\mathbf{H}_t^{(N)}$ is the t-th hidden state from the final Transformer layer (the t-th column of $\mathbf{H}^{(N)}$), and $m_t \in \{0,1\}$ is the indicator for padding.

We then project the global representation to obtain the final user embedding for user u, i.e.,

$$\mathbf{z}_u = \text{LayerNorm}(\mathbf{W}_g \mathbf{g}_u + \mathbf{b}_g) \tag{3.5}$$

where $\mathbf{W}_g \in \mathbb{R}^{d \times d}$ is a projection matrix, $\mathbf{b}_g$ is a bias vector, and LayerNorm$(\cdot)$ represents a normalization layer.

3.3.5 Masked Language Modelling for Semantic Embeddings

To learn semantic embeddings, we employ the masked language modelling (MLM) (Devlin et al., 2019). During training, we randomly mask a subset of items in each sequence and train the model to predict the masked items.

For a sequence S_u, we randomly select positions $M \subset \{1,2,...,L\}$ to mask with probability p_{mask}. The masked sequence $\tilde{S}_u$ is created by replacing items at positions in M with a special [MASK] token.

MLM learns semantic embedding by minimising the distance between the predicted masked items and the ground truth. The objective is computed as

$$L_{\text{MLM}} = -\frac{1}{|M|} \sum_{t \in M} \log p\left(i_t^u \mid \tilde{\mathbf{H}}_t^{(N)}\right) \tag{3.6}$$

where $\tilde{\mathbf{H}}_t^{(N)}$ is the hidden representation at position t from the masked sequence, and

$$p\left(i|\tilde{\mathbf{H}}_t^{(N)}\right) = \frac{\exp\left(\mathbf{w}_i^\top \tilde{\mathbf{H}}_t^{(N)}\right)}{\sum_{j=1}^{m} \exp\left(\mathbf{w}_j^\top \tilde{\mathbf{H}}_t^{(N)}\right)} \tag{3.7}$$

with $\mathbf{w}_i$ being the i-th row of the MLM prediction head $\mathbf{W}_{\text{MLM}} \in \mathbb{R}^{m \times d}$.

3.3.6 Partial Label Learning for Purchase Prediction

Since we lack explicit item acquisition labels during the model training phase, we use the idea of partial label learning to predict the item acquisition probability.

We maintain a partial label matrix $\bar{\mathbf{P}} = [\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2, \ldots, \bar{\mathbf{p}}_n] \in [0,1]^{m \times n}$ where $\bar{\mathbf{p}}_{u,i}$ indicates the probability that user u will purchase item i. We initialise partial labels $\bar{\mathbf{P}}$ by uniformly distributing probability mass across all items observed in each user's interaction sequence. Specifically, for each user u with interaction sequence $S_u = \{i_1^u, i_2^u, \ldots, i_L^u\}$, we set

$$\bar{\mathbf{p}}_{u,i}^{(0)} = \begin{cases} \frac{1}{|U_u|}, & \text{if } i \in U_u \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

where $U_u = \{i \mid i \in S_u\}$ represents the set of unique items in user u's interaction sequence.

Given a partial label matrix $\bar{\mathbf{P}}$, PLL uses the following objective computed using cross-entropy between the model predictions $\hat{\mathbf{p}}_u$ and the partial labels $\bar{\mathbf{P}}$ to update $\hat{\mathbf{p}}_u$, i.e.,

$$L_{\text{PLL}} = -\frac{1}{n} \sum_{u=1}^{n} \sum_{i=1}^{m} \bar{\mathbf{p}}_{u,i} \cdot \log(\hat{\mathbf{p}}_{u,i} + \epsilon) \tag{3.9}$$

where ϵ is a small constant (i.e., 10^{-10}) for numerical stability, $\hat{\mathbf{p}}_u^{(t)} = \text{softmax}\left(\mathbf{Z}_{\text{item}}^{\top} \mathbf{z}_u\right)$, $\mathbf{Z}_{\text{item}} \in \mathbb{R}^{d \times m}$ contains the item embeddings of all items.

PLL alternatively updates the partial labels $\bar{\mathbf{P}}$ and the model predictions $\hat{\mathbf{p}}_u$. Given the current model predictions $\hat{\mathbf{p}}_u^{(t)}$. The partial labels $\bar{\mathbf{P}}$ are then refined through a two-step process. In the first step,

$$\tilde{\mathbf{p}}_{u,i}^{(t+1)} = \begin{cases} \hat{\mathbf{p}}_{u,i}^{(t)} & \text{if } \bar{\mathbf{p}}_{u,i}^{(t)} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

Note that $\tilde{\mathbf{p}}_u^{(t+1)}$ is further normalised to maintain a valid probability distribution.

In the second step, to ensure stable training, we employ the momentum method, i.e.,

$$\bar{\mathbf{p}}_{u,i}^{(t+1)} = \alpha \cdot \tilde{\mathbf{p}}_{u,i}^{(t+1)} + (1-\alpha) \cdot \bar{\mathbf{p}}_{u,i}^{(t)} \tag{3.11}$$

where $\alpha \in [0,1]$ is the momentum coefficient that controls the trade-off between historical and current predictions.

3.3.7 Partial Acquisition Recommender System Algorithm

Optimising the objectives of MLM and PLL will enable the Transformer-based user embedding framework to learn meaningful user and item embeddings. Specifically, MLM will help capture sequential information. With the assumption that the majority of sequences contain at least one item to purchase, PLL will help extract reliable item acquisition information, i.e., $\hat{\mathbf{p}}_{u,i}$ will be large if the user u is likely to purchase the item i and $\hat{\mathbf{p}}_{u,i}$ will be small in the other way. We therefore build the objective of Partial Acquisition Recommender Systems (PARS) as follows:

$$L = L_{\text{PLL}} + \lambda \cdot L_{\text{MLM}} \tag{3.12}$$

where λ is a hyperparameter controlling the weight of the MLM objective. The complete training procedure is summarised in Algorithm 1.

At the inference phase, given a user's interaction sequence $S_{u_{\text{new}}}$, we can use the trained Transformer encoder to obtain the user embedding $\mathbf{z}_{u_{\text{new}}}$. We then can calculate $\hat{\mathbf{p}}_{u_{\text{new}}} = \mathbf{z}_{u_{\text{new}}}^{\top} \mathbf{Z}_{\text{item}}$. PARS ranks the items according to $\hat{\mathbf{p}}_{u_{\text{new}}}$ to generate purchase recommendations.

The key advantage of our approach is that it can learn meaningful purchase patterns through self-supervision and partial label learning, without requiring explicit item acquisition labels during training. This makes it applicable to a wide range of real-world scenarios where obtaining complete item acquisition labels is challenging or impossible.

Algorithm 1 Partial Acquisition Recommender Systems

Require: User sequences $\{S_u\}$, number of epochs T, learning rate η, MLM weight λ
Ensure: Trained model parameters θ

Initialize model parameters θ randomly
Initialize partial labels $\mathbf{Y}^{(0)}$
for epoch = 1 to T **do**
  for each batch of sequences **do**
    // *Masked Language Modeling*
    Create masked sequences
    Compute sequence representations $\mathbf{H}$
    Compute L_{MLM} using masked positions
    // *Partial Label Learning*
    Compute user embedding $\mathbf{z}_u$
    Compute prediction scores $\hat{\mathbf{p}}_u$
    Update partial labels $\bar{\mathbf{P}}$
    Compute L_{PLL} using updated partial labels
    // *Parameter Update*
    $L = L_{\text{PLL}} + \lambda \cdot L_{\text{MLM}}$
    $\theta = \theta - \eta \cdot \nabla_\theta L$
  end for
end forθ =0

3.4 Experiments

3.4.1 Datasets

To comprehensively evaluate our proposed method, we conduct extensive experiments on three large-scale public datasets, i.e., YooChoose, RetailRocket, and Taobao, collected from real-world e-commerce platforms. These datasets represent diverse user behaviours and purchase patterns across different platforms and regions.

Yoochoose. The Yoochoose dataset[4] is collected from an e-commerce website and contains click-stream data from a six-month period. The dataset includes 9.25 million sessions with 52.74 million clicks on 52,739 unique items. We follow the standard preprocessing protocol by filtering out sessions with only one interaction and items that appear less than 5 times. The preprocessed dataset contains approximately 6.2 million

[4] `https://www.kaggle.com/chadgostopp/recsys-challenge-2015`

Table 3.1: Statistics of the experimental datasets.

Dataset	#Users	#Items	#Interactions	Avg. Length
YooChoose	417,370	39,299	6,197,045	14.85
RetailRocket	116,679	112,574	620,338	5.32
Taobao	213,374	68,777	3,563,298	16.70

sessions and 39,299 items.

RetailRocket. The RetailRocket dataset[5] is collected from a real-world e-commerce website over a period of 4.5 months. It contains user behaviour data, including views, add-to-cart events, and transaction data. The dataset comprises 2.76 million events from 1.41 million unique visitors interacting with 235,061 items. After preprocessing to remove items with fewer than 5 interactions, we obtain 0.6 million sessions with 112,574 unique items.

Taobao. The Taobao dataset[6] is provided by Alibaba and contains user behaviour logs from the Taobao mobile application. The dataset includes various types of user-item interactions such as clicks, purchases, add-to-cart, and add-to-favourites over an 8-day period. It contains 100 million user behaviours from 987,994 users on 4.16 million items. After preprocessing to remove items with fewer than 5 interactions, resulting of 213,374 users and 68,777 items with 3.56 million interactions.

Table 3.1 summarises the statistics of the three datasets after preprocessing. The datasets vary significantly in scale and characteristics, with YooChoose having the most sessions but fewer items, RetailRocket being more sparse with shorter sequences, and Taobao containing rich user behaviour patterns with longer interaction sequences.

[5] https://www.kaggle.com/retailrocket/ecommerce-dataset
[6] https://tianchi.aliyun.com/dataset/649?lang=en-us

3.4.2 Evaluation Metrics

We adopt a comprehensive set of evaluation metrics to assess the performance of our method from multiple perspectives:

AUC (Area Under the ROC Curve): AUC measures the overall ranking quality by computing the probability that a randomly chosen positive item is ranked higher than a randomly chosen negative item. It is particularly suitable for imbalanced datasets common in purchase prediction tasks.

Hit Rate@k (HR@k): HR@k calculates the proportion of test cases where at least one relevant item appears in the top-k recommendations. It is defined as:

$$\text{HR@k} = \frac{1}{|U|} \sum_{u \in U} \mathbb{I}(|R_u \cap P_u^k| > 0) \tag{3.13}$$

where R_u is the set of relevant items for user u, P_u^k is the set of top-k predicted items, and $\mathbb{I}(\cdot)$ is the indicator function.

NDCG@k (Normalized Discounted Cumulative Gain): NDCG@k measures the ranking quality by considering the position of relevant items in the recommendation list:

$$\text{NDCG@k} = \frac{1}{|U|} \sum_{u \in U} \frac{\text{DCG}_u^k}{\text{IDCG}_u^k} \tag{3.14}$$

where DCG and IDCG represent the discounted cumulative gain and ideal discounted cumulative gain, respectively.

Precision@k: Precision@k measures the fraction of relevant items among the top-k recommendations:

$$\text{Precision@k} = \frac{1}{|U|} \sum_{u \in U} \frac{|R_u \cap P_u^k|}{k} \tag{3.15}$$

Recall@k: Recall@k measures the fraction of relevant items that are successfully retrieved in the top-k recommendations:

$$\text{Recall@k} = \frac{1}{|U|} \sum_{u \in U} \frac{|R_u \cap P_u^k|}{|R_u|} \tag{3.16}$$

F1@k: F1@k is the harmonic mean of Precision@k and Recall@k, providing a balanced measure:

$$\text{F1@k} = 2 \cdot \frac{\text{Precision@k} \cdot \text{Recall@k}}{\text{Precision@k} + \text{Recall@k}} \tag{3.17}$$

We evaluate all metrics with $k \in \{1, 3, 5\}$ to assess performance at different recommendation list lengths.

3.4.3 Baseline Methods

Note that PARS is the first methodology to enable building recommender systems only using browsing histories; there is no straightforward baseline. We compare PARS against ten state-of-the-art CVR baseline methods that using item acquisition labels.

- **DeepFM** (Guo et al., 2017): Combines factorisation machines with deep neural networks to capture both low-order and high-order feature interactions.
- **ESMM** (Ma et al., 2018b): Entire Space Multi-Task Model addresses sample selection bias by modelling CVR over entire exposure space rather than clicked samples.
- **MMoE** (Ma et al., 2018a): Multi-gate Mixture-of-Experts learns task-specific and shared representations through a gating mechanism for multi-task learning.
- **BERT4Rec** (Sun et al., 2019a): Bidirectional Encoders from Transformer for SR that uses bidirectional self-attention to model user behaviour sequences and employs Cloze task for pre-training to capture contextual information from both directions.
- **ESM**^{2} (Wen et al., 2020): Entire Space Multi-task Modeling with field-level calibration to handle task conflicts and improve CVR prediction.

- **ESCM**^{2} (Wang et al., 2022a): Entire Space Counterfactual Multi-task Modelling that employs counterfactual risk minimisation to address selection bias and data sparsity.

- **DCMT** (Zhu et al., 2023): Disentangled Causal Multi-Task learning that separately models CVR in click and non-click spaces to reduce gradient conflicts.

- **CL4CVR** (Ouyang et al., 2023): Contrastive Learning for CVR prediction that leverages self-supervised learning to improve representation learning.

- **DDPO** (Su et al., 2024): Doubly-robust estimator with Dual Propensity Optimisation that combines inverse propensity weighting with direct prediction.

- **NISE** (Huang et al., 2024a): Negative sampling with Importance Sampling Estimator that addresses exposure bias through importance-weighted learning.

- **EVI** (Fei et al., 2025): An entire-space variational information exploitation framework that generates unbiased pseudo labels via a conditional CVR teacher and transfers non-click information using variational inference and logit distillation to improve CVR prediction.

These baseline methods represent the current state-of-the-art in CVR prediction, covering diverse approaches that include multi-task learning, causal inference, deep feature interaction modelling, and various debiasing techniques. Compared against this comprehensive set of baselines, we can thoroughly evaluate the effectiveness of PARS. Our implementation is publicly available at the following link[7].

[7] `https://github.com/bjtu-lucas-nlp/PARS`

3.4.4 Implementation Details

3.4.4.1 Experimental Setup

All experiments were conducted on a server equipped with 2 NVIDIA A100 GPUs (80GB memory each), Intel(R) Xeon(R) Gold 6342 CPU@2.80GHz, and 512GB RAM. We implemented all models using PyTorch 2.4.1 with CUDA 12.1.

3.4.4.2 Model Configurations

We employ consistent architectural choices across all models for fair comparison. All models use batch normalisation, Xavier initialisation, and gradient clipping with a maximum norm of $1 - 10^{-15}$, and a minimum norm of 10^{-15}. The maximum sequence length is set to 50 for all datasets.

PARS (Ours). We employ a two-tower architecture with embedding dimension $d = 128$, mask ratio of 0.3, and dropout rate of 0.2. The model is optimised using Adam with learning rate $\eta = 10^{-3}$, batch size 256, and trained for 300 epochs. The task balancing weight λ is set to 1.0.

Baseline Models. We configure all baseline models with the following detailed key parameters:

- **DeepFM:** Embedding size 128, MLP layers [128, 64], dropout 0.2, and learning rate 0.001
- **ESMM:** Shared embedding dimension 64, task-specific towers [64, 32], dropout 0.2, and learning rate 0.001
- **MMOE:** 3 expert networks with [128, 64] units each, embedding size 64, dropout 0.2, and learning rate 0.001
- **BERT4Rec:** We adopt a bidirectional Transformer-based sequential recommendation architecture. The embedding dimension is set to 64, and the encoder contains 2

Transformer layers, each with 2 attention heads. A GELU activation and residual layer normalisation follow each attention block. The hidden dropout rate is 0.2, and the attention dropout rate is 0.1. The maximum sequence length is 100, and the masking ratio is fixed to 15%, following the original BERT4Rec training objective. The model is optimised using Adam with a learning rate of 0.001.

- **ESM^2:** Shared embedding dimension 64, dropout 0.2, two layers in the MLP are set, where the dimension of each layer is set to 128, 64, 32, and learning rate 0.001
- **ESCM^2-DR:** Counterfactual tower [64, 32], dropout 0.2, and learning rate 0.001
- **ESCM^2-IPS:** Counterfactual tower [64, 32], dropout 0.2, and learning rate 0.001
- **DCMT:** Counterfactual tower [64, 32], dropout 0.2, and learning rate 0.001
- **CL4CVR:** Contrastive temperature $\tau = 0.1$, embedding dimension 64, dropout 0.2, and learning rate 0.001
- **DDPO:** Counterfactual tower [64, 32], dropout 0.2, and learning rate 0.001
- **NISE:** Counterfactual tower [64, 32], dropout 0.2, and learning rate 0.001
- **EVI:** Expert num 3, embedding dimension 64, dropout 0.2, and learning rate 0.001

3.4.4.3 Training Details

Models were trained until a maximum of 300 epochs. We employ log loss for conversion prediction. All hyperparameters were tuned on the validation set using Adam optimisation.

3.4.5 Impact of Label Availability on Baseline Performance

We investigate how label availability affects baseline model performance. Ablation studies are conducted by providing baseline training models with varying ratios of

ground-truth labels. Note that when the ratio equals 50%, it means that the baseline methods use 50% of the positive acquisition labels and set the rest positive acquisition labels to zeros. Figure 3.3 illustrates the performances of DeepFM using different label ratios on all three datasets. Remarkably, PARS without using item acquisition labels outperforms DeepFM using 100% item acquisition labels.

We further conduct comprehensive analyses examining the impact of varying training label proportions on the 10 baseline algorithms across three datasets under different evaluation metrics. We compare these results against our PARS algorithm, which operates without any acquisition labels during training. As illustrated in Figures 4-14, we observe that current baseline algorithms exhibit high sensitivity to the quantity of acquisition labels. Notably, when limited to small label proportions (e.g., less than 10%), baseline performance degrades dramatically. This observation underscores the practical significance of our PARS algorithm, particularly in real-world recommendation scenarios where acquisition labels are scarce or costly to obtain.

The results reveal a critical finding: baseline performance deteriorates significantly when label availability drops below 10%, with AUC scores declining by up to 35% on YooChoose and 42% on Taobao. This dramatic performance drop underscores the heavy reliance of traditional CVR models on labelled data.

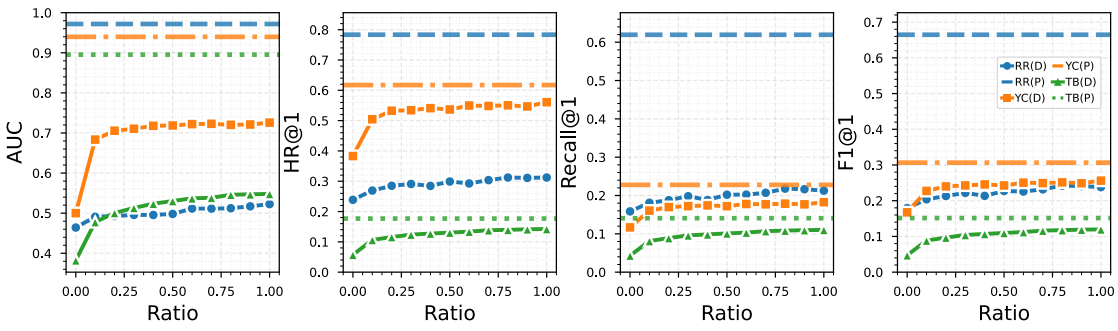

Figure 3.3: Performance comparison across multiple evaluation metrics with varying the ratios of using positive acquisition labels on three e-commerce datasets. (Abbreviations: RR=RetailRocket, YC=YooChoose, TB=Taobao, D=DeepFM, P=PARS.)

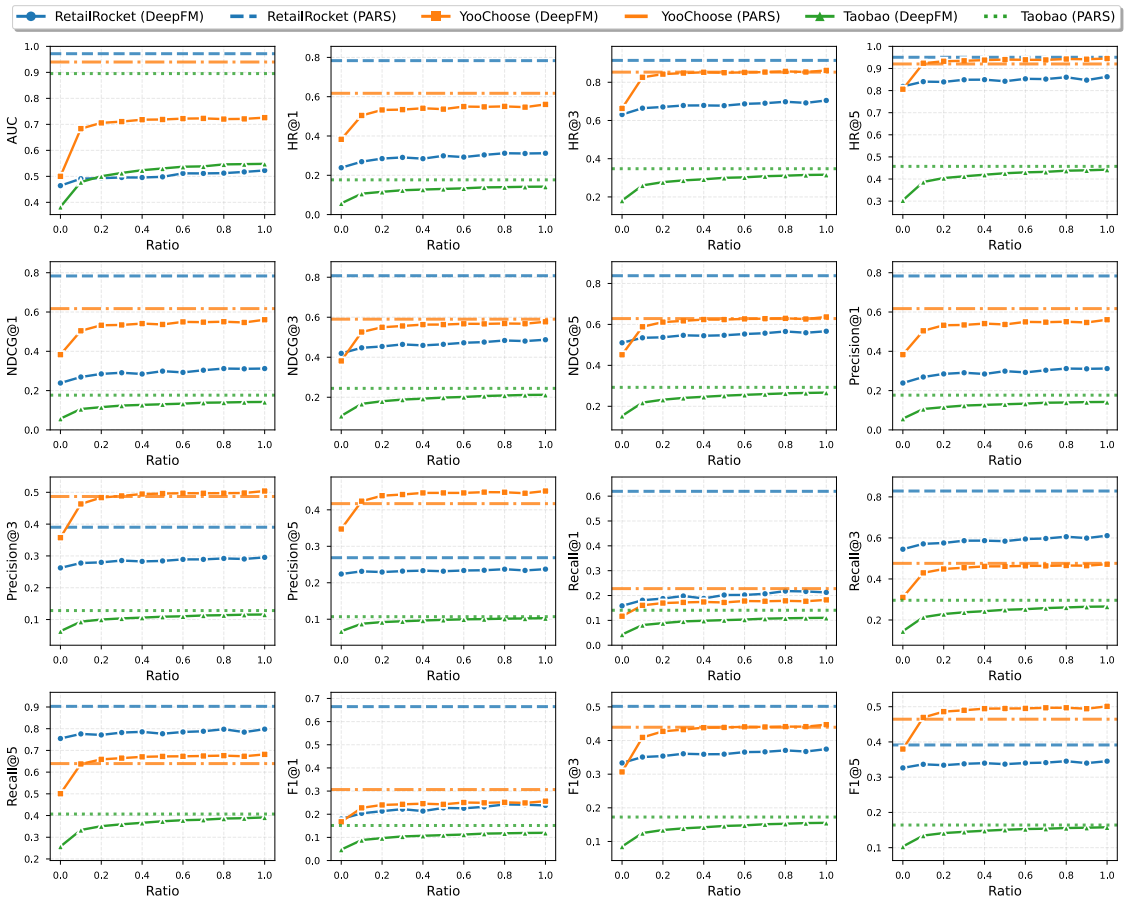

Figure 3.4: Performance comparison of DeepFM across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

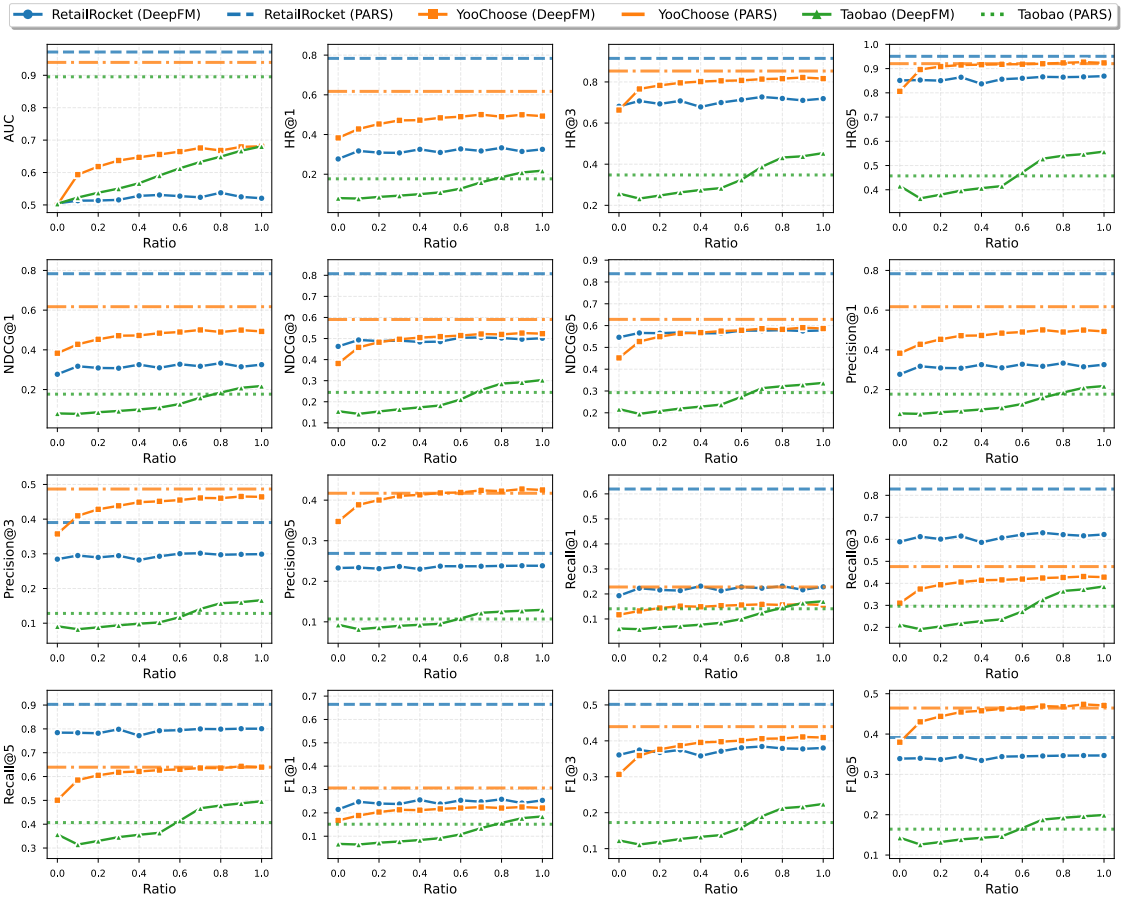

Figure 3.5: Performance comparison of ESMM across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

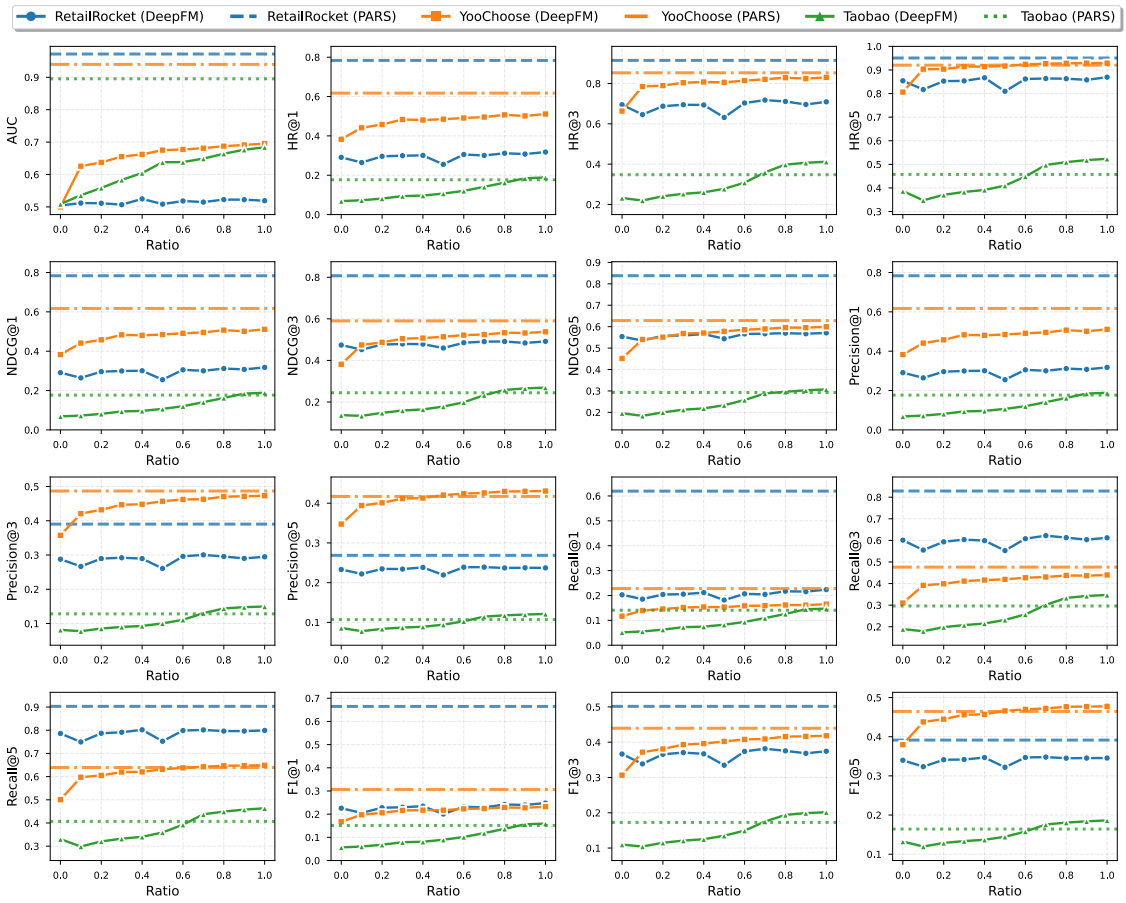

Figure 3.6: Performance comparison of MMOE across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

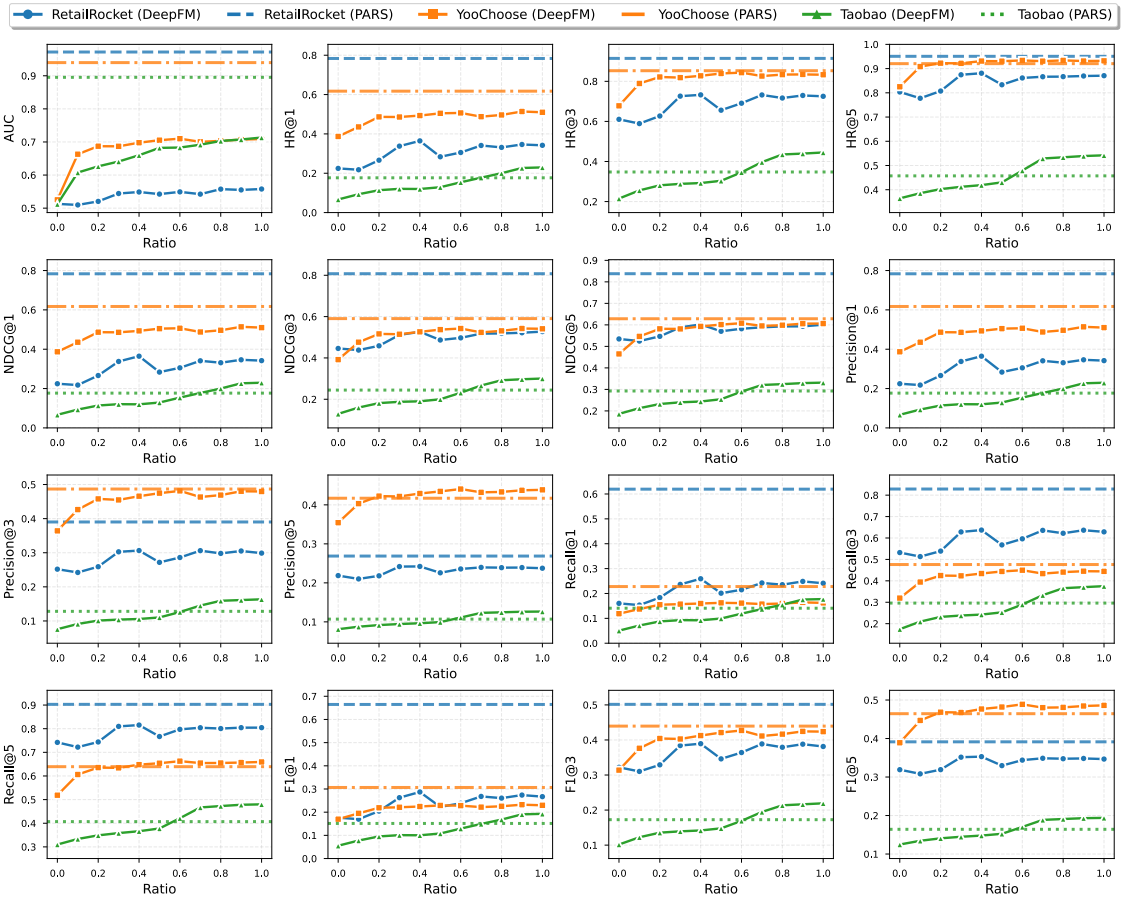

Figure 3.7: Performance comparison of ESM^2 across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

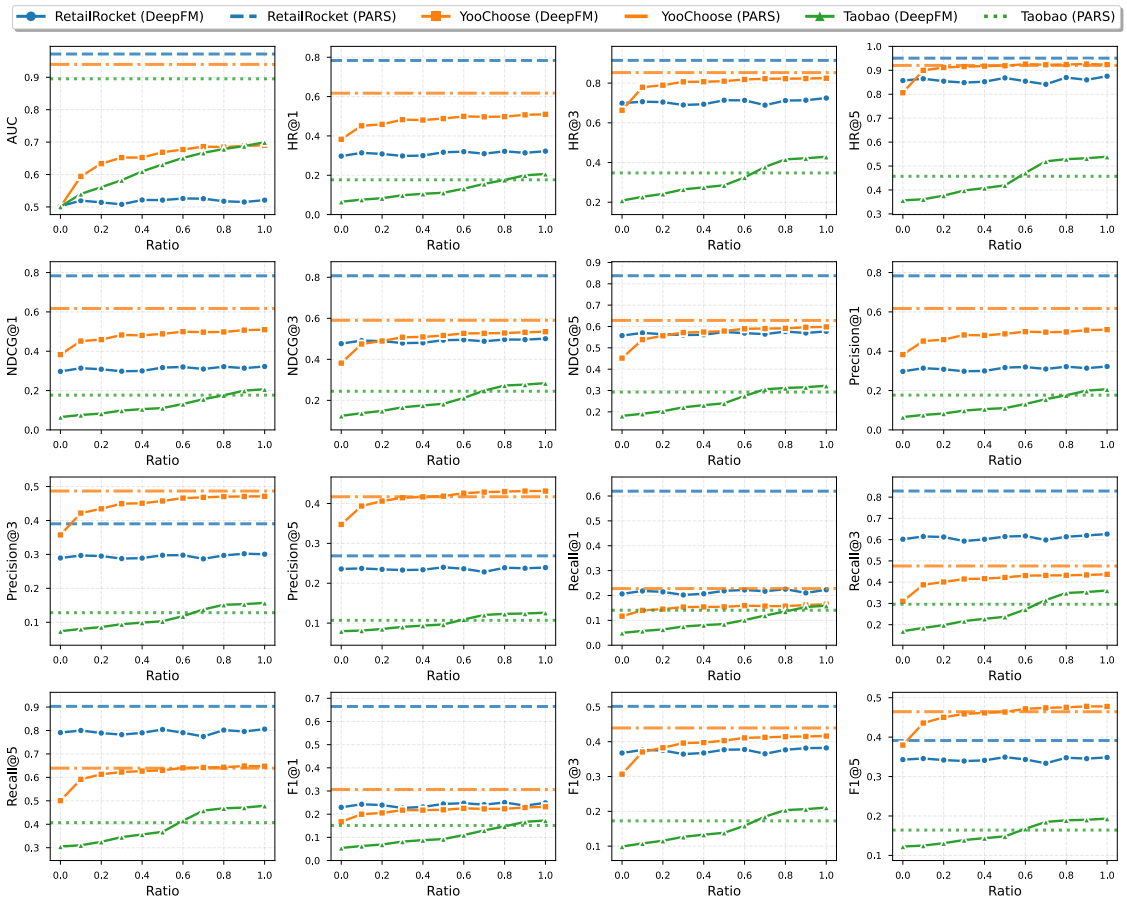

Figure 3.8: Performance comparison of ESCM^2-DR across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

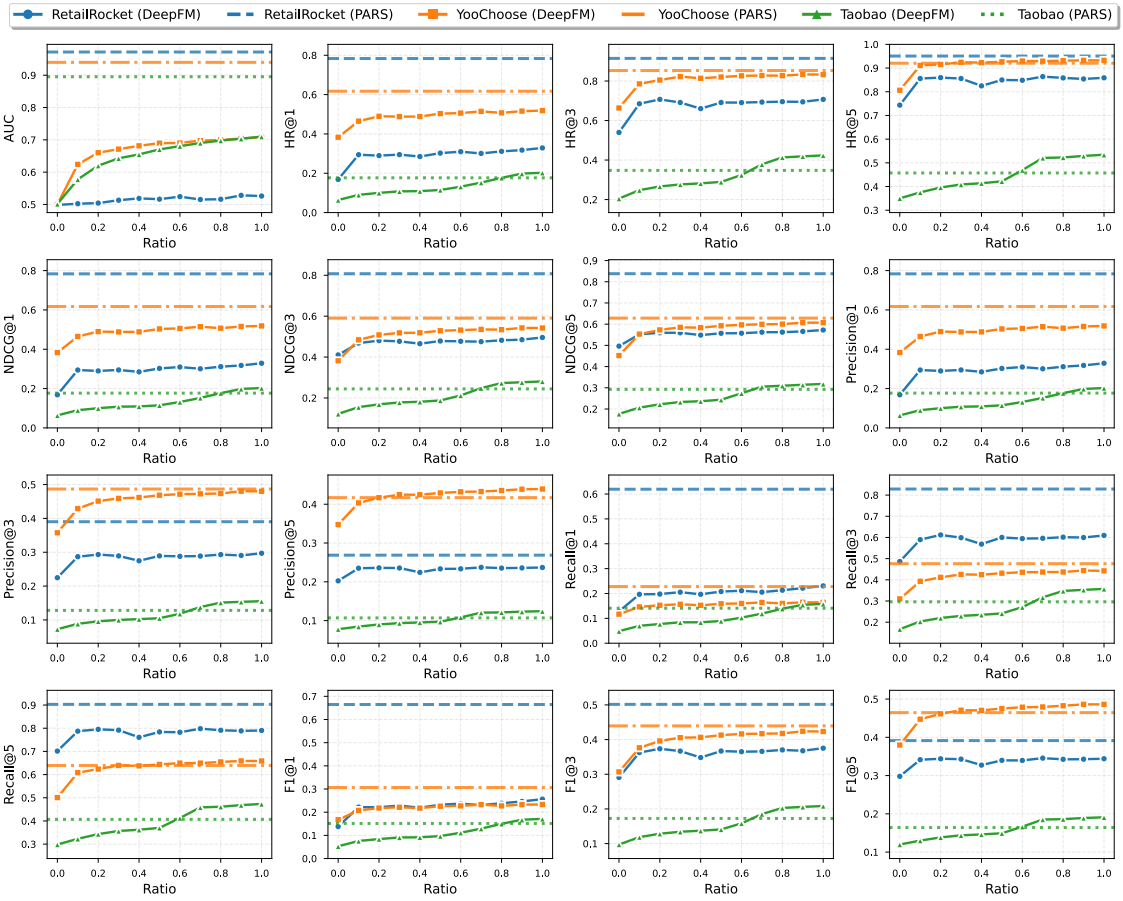

Figure 3.9: Performance comparison of ESCM^2-IPS across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

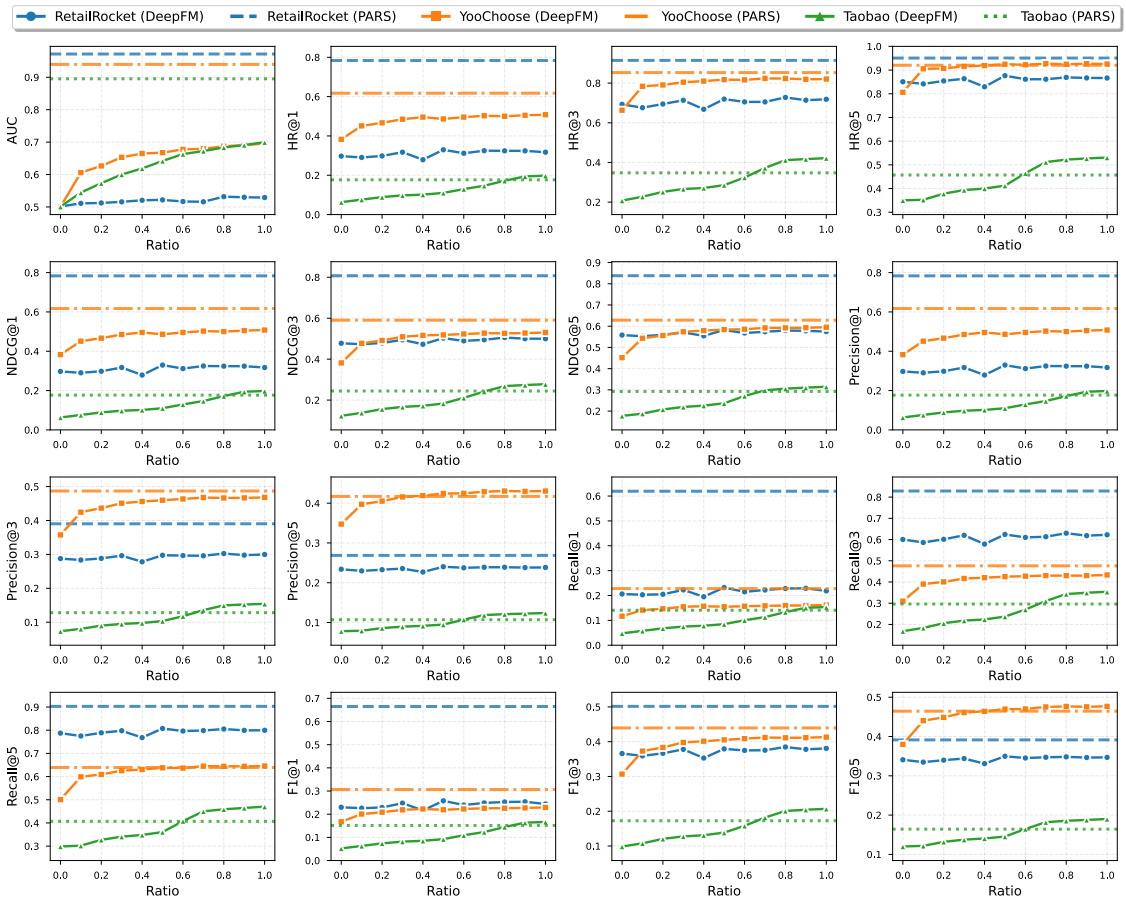

Figure 3.10: Performance comparison of DCMT across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

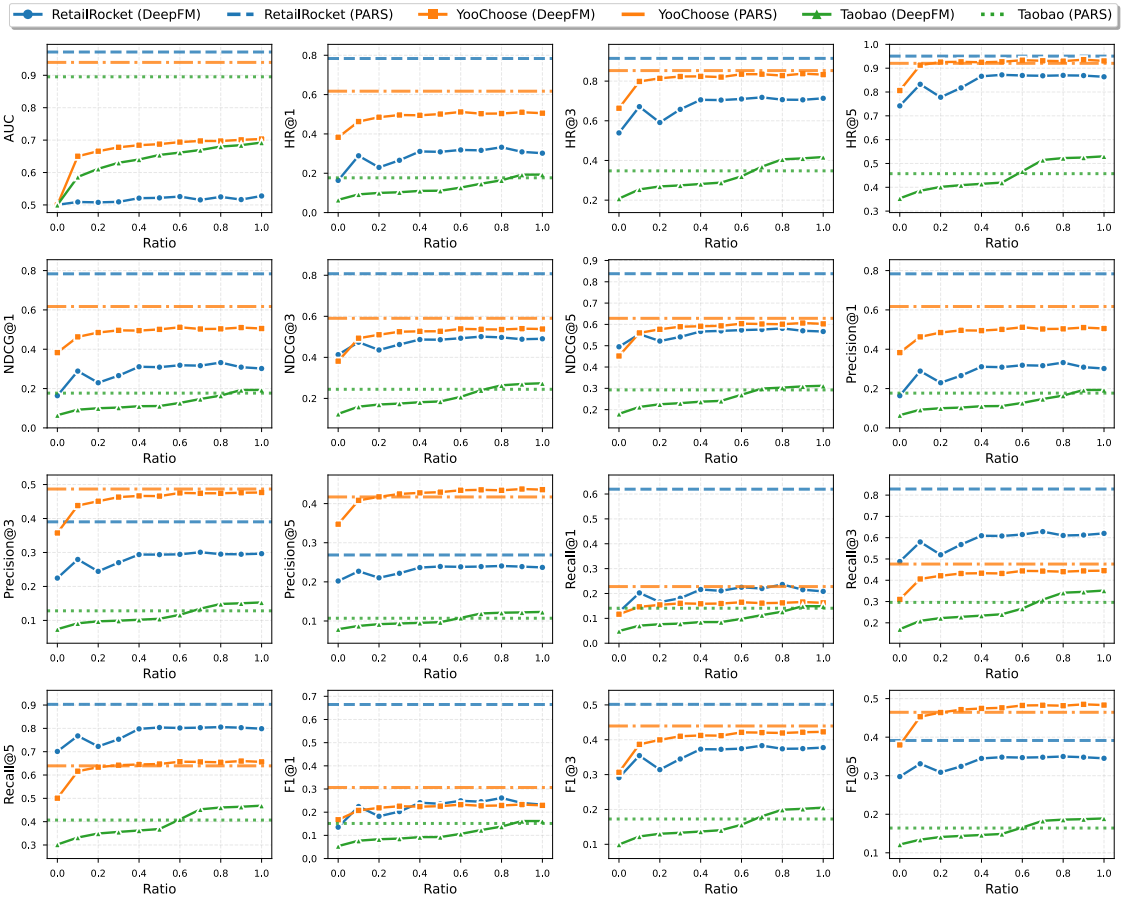

Figure 3.11: Performance comparison of CL4CVR across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

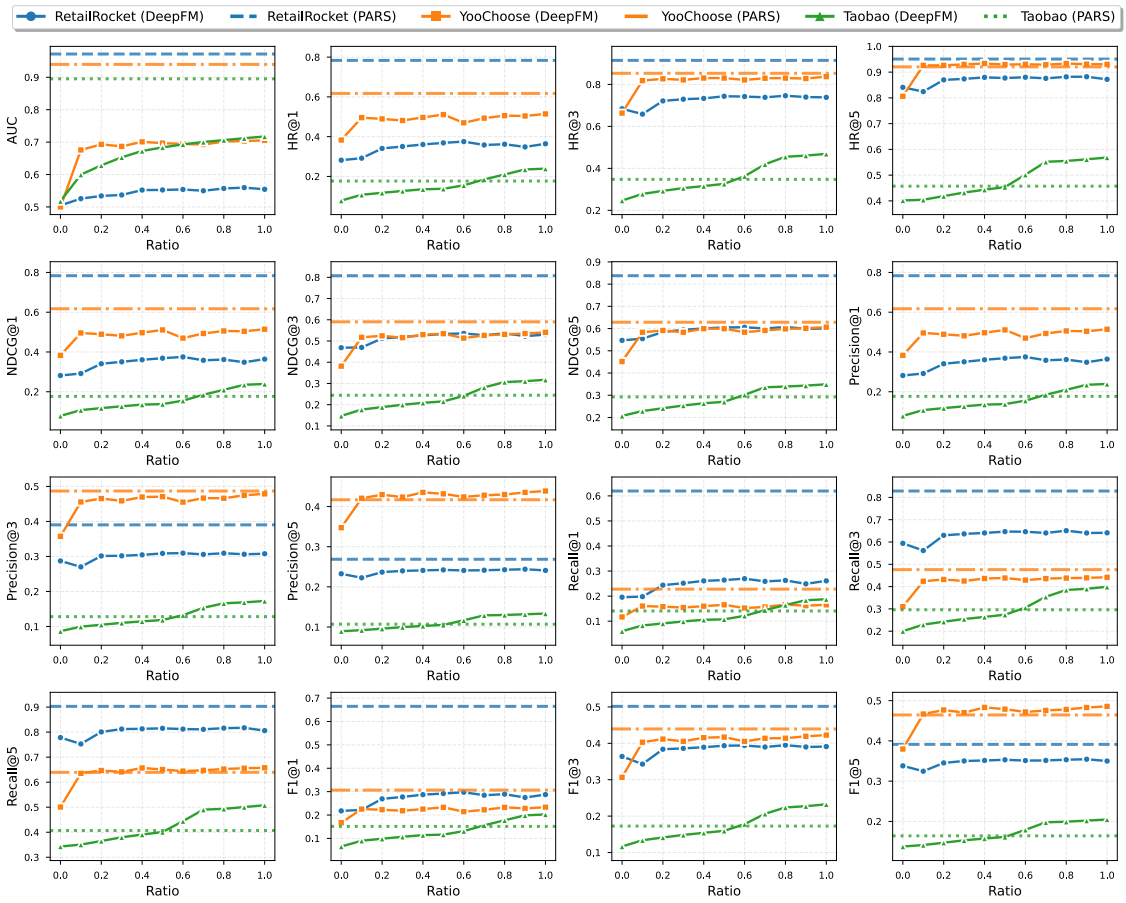

Figure 3.12: Performance comparison of DDPO across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

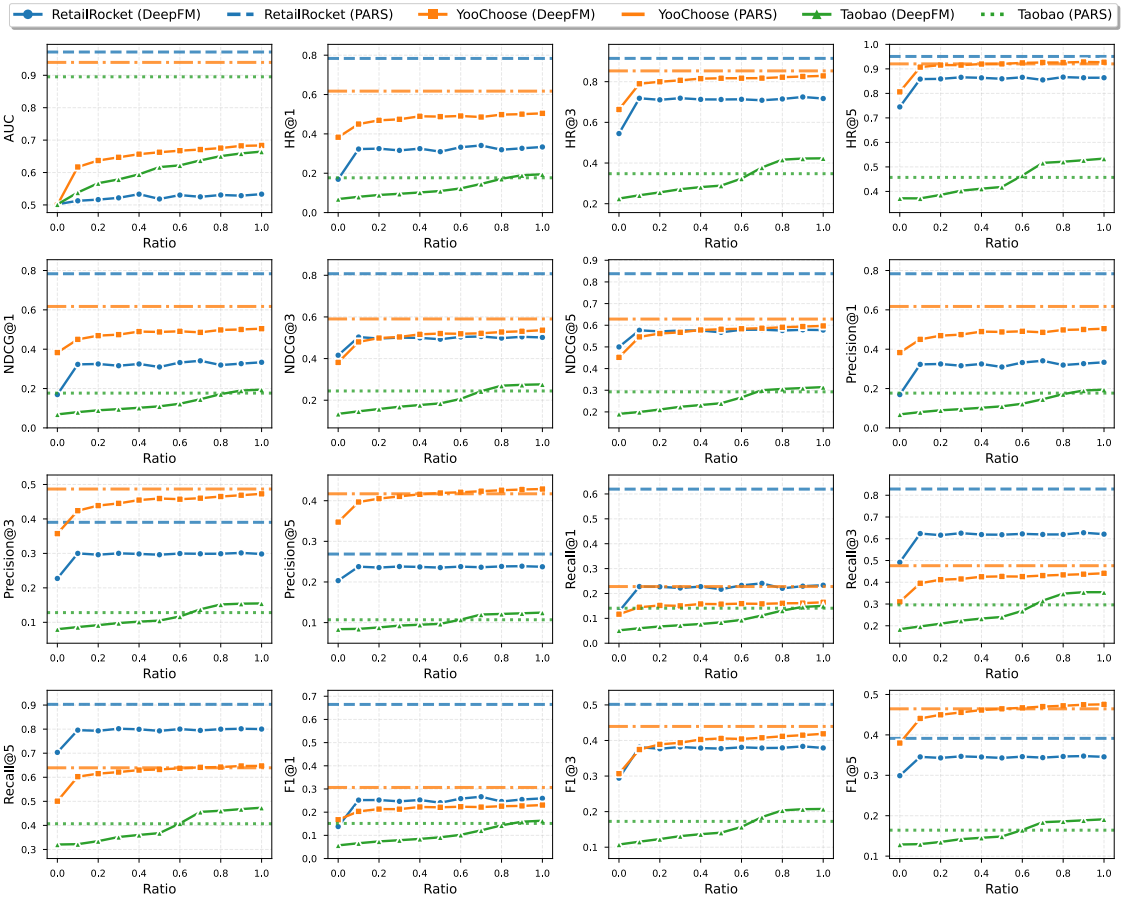

Figure 3.13: Performance comparison of NISE across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

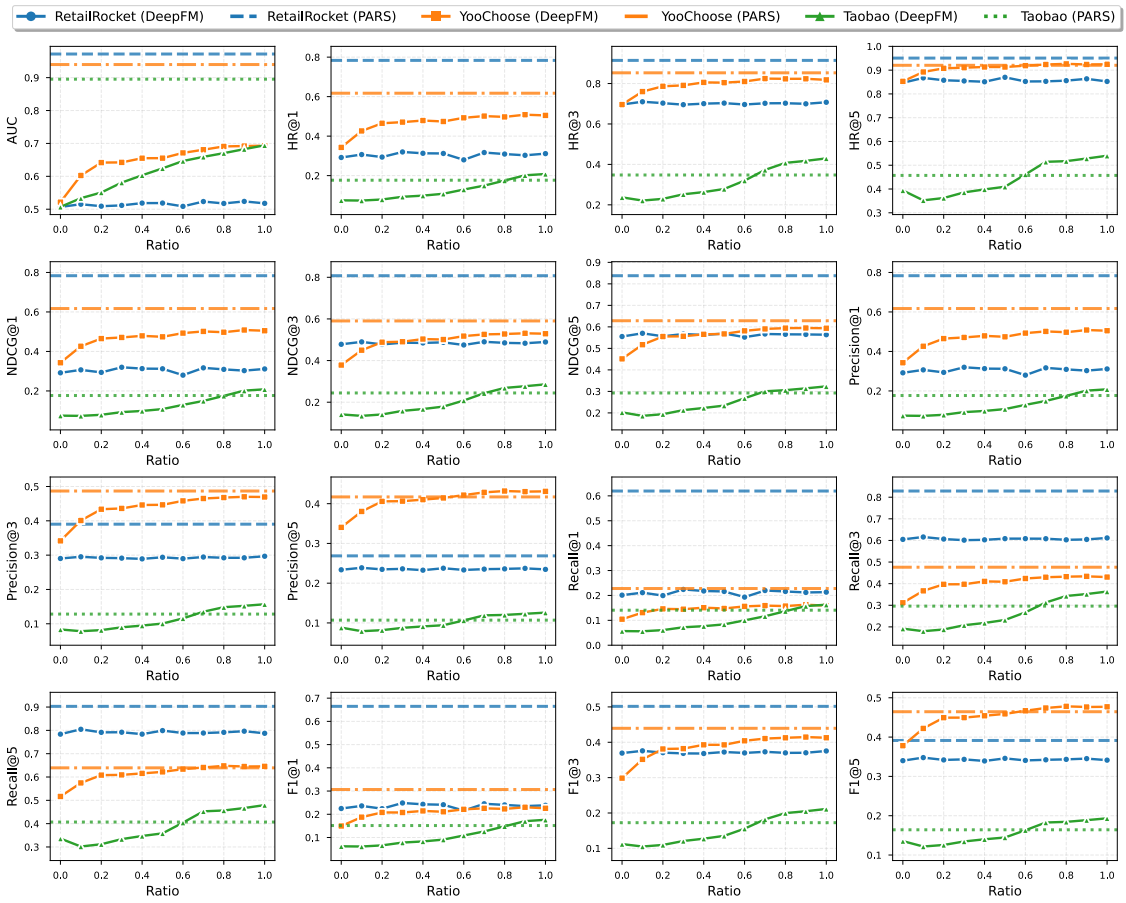

Figure 3.14: Performance comparison of EVI across multiple evaluation metrics with varying positive-negative sampling ratios on three e-commerce datasets.

Table 3.2: Performance comparison between our zero-shot approach (0% labels) and fully-supervised baselines (50% labels) across three datasets. Best results are in **bold**, second best are underlined.

Dataset	Method	AUC	HR@K			NDCG@K		Precision@K		Recall@K			F1@K		
			@1	@3	@5	@3	@5	@3	@5	@1	@3	@5	@1	@3	@5
YooChoose	DeepFM	0.6964	0.5379	0.8334	0.9085	0.5438	0.6053	0.4768	0.4140	0.1693	0.4381	0.6299	0.2407	0.4193	0.4599
	ESMM	0.5970	0.5016	0.8164	0.9017	0.5226	0.5881	0.4605	0.4045	0.1611	0.4247	0.6181	0.2273	0.4060	0.4501
	MMOE	0.6724	0.4870	0.8108	0.8994	0.5132	0.5767	0.4555	0.3982	0.1547	0.4188	0.6083	0.2191	0.4008	0.4429
	BERT4Rec	0.5786	0.4066	0.7332	0.8741	0.4325	0.5049	0.3875	0.3690	0.1289	0.3590	0.5641	0.1822	0.3412	0.4104
	ESM2	0.6945	0.5070	0.8280	0.9065	0.5334	0.5968	0.4721	0.4119	0.1616	0.4359	0.6282	0.2284	0.4163	0.4583
	$ESCM^2$-DR	0.6327	0.4926	0.8199	0.9054	0.5193	0.5867	0.4610	0.4067	0.1557	0.4238	0.6214	0.2210	0.4058	0.4529
	$ESCM^2$-IPS	0.6621	0.5063	0.8282	0.9096	0.5287	0.5943	0.4670	0.4094	0.1598	0.4325	0.6268	0.2271	0.4125	0.4562
	DCMT	0.6687	0.4944	0.8211	0.9023	0.5226	0.5866	0.4640	0.4060	0.1556	0.4288	0.6194	0.2213	0.4094	0.4519
	CL4CVR	0.6635	0.5006	0.8278	0.9102	0.5269	0.5935	0.4653	0.4091	0.1606	0.4341	0.6279	0.2270	0.4122	0.4564
	DDPO	0.6838	0.5026	0.8274	0.9063	0.5244	0.5890	0.4617	0.4043	0.1614	0.4289	0.6211	0.2277	0.4082	0.4511
	NISE	0.5946	0.5012	0.8256	0.9116	0.5260	0.5909	0.4616	0.4045	0.1624	0.4329	0.6235	0.2283	0.4095	0.4516
	EVI	0.6597	0.4888	0.8177	0.9044	0.5177	0.5826	0.4595	0.4028	0.1545	0.4241	0.6170	0.2191	0.4052	0.4489
	PARS (Ours)	**0.9398**	**0.6173**	**0.8530**	**0.9204**	**0.5903**	**0.6285**	**0.4868**	**0.4170**	**0.2278**	**0.4761**	**0.6393**	**0.3066**	**0.4395**	**0.4645**
RetailRocket	DeepFM	0.5061	0.2965	0.6852	0.8499	0.4702	0.5512	0.2884	0.2344	0.1980	0.5934	0.7847	0.2231	0.3647	0.3407
	ESMM	0.5188	0.3207	0.7146	0.8707	0.4978	0.5769	0.2987	0.2385	0.2242	0.6184	0.8039	0.2492	0.3791	0.3477
	MMOE	0.5166	0.3063	0.7023	0.8503	0.4813	0.5584	0.2899	0.2336	0.2121	0.6048	0.7846	0.2365	0.3691	0.3402
	BERT4Rec	0.5103	0.3156	0.7244	0.8664	0.4996	0.5732	0.2994	0.2363	0.2198	0.6289	0.8011	0.2452	0.3828	0.3454
	ESM2	0.5487	0.3650	0.7376	0.8792	0.5311	0.6040	0.3085	0.2427	0.2612	0.6453	0.8160	0.2884	0.3929	0.3532
	$ESCM^2$-DR	0.5156	0.3207	0.7027	0.8562	0.4936	0.5723	0.2934	0.2340	0.2193	0.6077	0.7893	0.2455	0.3717	0.3410
	$ESCM^2$-IPS	0.5217	0.3041	0.6891	0.8537	0.4762	0.5581	0.2874	0.2350	0.2084	0.5961	0.7873	0.2331	0.3647	0.3418
	DCMT	0.5246	0.3271	0.6976	0.8677	0.4917	0.5753	0.2921	0.2373	0.2290	0.6047	0.8011	0.2549	0.3707	0.3459
	CL4CVR	0.5179	0.3131	0.6937	0.8541	0.4850	0.5659	0.2914	0.2352	0.2203	0.6018	0.7909	0.2444	0.3693	0.3427
	DDPO	0.5468	0.3764	0.7350	0.8796	0.5357	0.6078	0.3061	0.2393	0.2777	0.6425	0.8127	0.3037	0.3907	0.3498
	NISE	0.5265	0.3296	0.7095	0.8673	0.4988	0.5789	0.2956	0.2390	0.2287	0.6169	0.8043	0.2545	0.3760	0.3482
	EVI	0.5079	0.3271	0.7091	0.8562	0.4955	0.5723	0.2934	0.2354	0.2306	0.6108	0.7910	0.2556	0.3729	0.3426
	PARS (Ours)	**0.9718**	**0.7835**	**0.9145**	**0.9507**	**0.8076**	**0.8379**	**0.3902**	**0.2687**	**0.6194**	**0.8286**	**0.9030**	**0.6647**	**0.5016**	**0.3914**
Taobao	DeepFM	0.5303	0.1305	0.2999	0.4261	0.1983	0.2519	0.1089	0.0983	0.1010	0.2497	0.3736	0.1097	0.1461	0.1508
	ESMM	0.5043	0.1088	0.2838	0.4152	0.1823	0.2376	0.1023	0.0952	0.0847	0.2366	0.3640	0.0918	0.1377	0.1464
	MMOE	0.5044	0.1068	0.2777	0.4097	0.1781	0.2333	0.1003	0.0941	0.0824	0.2317	0.3591	0.0896	0.1349	0.1445
	BERT4Rec	0.5039	0.1020	0.2767	0.4168	0.1758	0.2350	0.1004	0.0964	0.0783	0.2303	0.3670	0.0853	0.1347	0.1480
	ESM2	0.5282	0.1289	0.3036	0.4304	0.1999	0.2543	0.1103	0.0994	0.0993	0.2529	0.3779	0.1080	0.1479	0.1525
	$ESCM^2$-DR	0.5127	0.1109	0.2853	0.4192	0.1831	0.2401	0.1030	0.0965	0.0850	0.2371	0.3677	0.0925	0.1384	0.1482
	$ESCM^2$-IPS	0.5128	0.1148	0.2893	0.4218	0.1876	0.2436	0.1050	0.0971	0.0893	0.2413	0.3700	0.0968	0.1409	0.1492
	DCMT	0.5093	0.1102	0.2857	0.4122	0.1833	0.2368	0.1033	0.0947	0.0847	0.2370	0.3605	0.0922	0.1386	0.1454
	CL4CVR	0.5106	0.1112	0.2884	0.4197	0.1852	0.2411	0.1048	0.0970	0.0851	0.2403	0.3684	0.0927	0.1405	0.1489
	DDPO	0.5626	0.1378	0.3262	0.4536	0.2154	0.2705	0.1186	0.1051	0.1073	0.2739	0.4007	0.1162	0.1595	0.1614
	NISE	0.5068	0.1096	0.2889	0.4182	0.1848	0.2400	0.1050	0.0969	0.0841	0.2405	0.3679	0.0916	0.1408	0.1486
	EVI	0.5061	0.1081	0.2784	0.4095	0.1789	0.2339	0.1006	0.0942	0.0837	0.2320	0.3583	0.0909	0.1353	0.1446
	PARS (Ours)	**0.8953**	**0.1768**	**0.3476**	**0.4572**	**0.2444**	**0.2926**	**0.1282**	**0.1070**	**0.1407**	**0.2961**	**0.4069**	**0.1514**	**0.1726**	**0.1642**

3.4.5.1 Overall Comparative Analysis

Although PARS outperforms most CVR baselines using 100% item acquisition labels, its performance on Taobao is not significant because Taobao has a very high number of positive item acquisition labels, i.e., 75.45% of the user interaction sequences contain at least one item marked as purchased. Table 3.2 shows that, under different metrics, PARS without using item acquisition labels remarkably outperforms all baselines using 50% positive item acquisition labels, showing the effectiveness of PARS.

In Table 3.3, we show that PARS without using any item acquisition labels outperforms most of the CVR baselines using 100% item acquisition labels. Our PARS algorithm

Table 3.3: Performance comparison between our zero-shot approach (0% labels) and fully-supervised baselines (100% labels) across three datasets. Best results are in **bold**, second best are underlined.

Dataset	Method	AUC	HR@K			NDCG@K		Precision@K		Recall@K			F1@K		
			@1	@3	@5	@3	@5	@3	@5	@1	@3	@5	@1	@3	@5
YooChoose	DeepFM	0.7258	0.5607	**0.8619**	**0.9450**	0.5773	**0.6365**	0.5040	**0.4516**	0.1825	0.4718	**0.6817**	0.2562	**0.4472**	**0.5010**
	ESMM	0.6808	0.4928	0.8162	0.9236	0.5231	0.5865	0.4642	0.4251	0.1559	0.4282	0.6394	0.2209	0.4092	0.4709
	MMOE	0.6954	0.5112	0.8301	0.9289	0.5376	0.5995	0.4732	0.4306	0.1650	0.4398	0.6487	0.2324	0.4183	0.4775
	BERT4Rec	0.5786	0.4066	0.7332	0.8741	0.4325	0.5049	0.3875	0.3690	0.1289	0.3590	0.5641	0.1822	0.3412	0.4104
	ESM^2	0.7088	0.5099	0.8330	0.9321	0.5410	0.6061	0.4796	0.4385	0.1621	0.4442	0.6593	0.2297	0.4238	0.4861
	$ESCM^2$-DR	0.6903	0.5097	0.8247	0.9236	0.5347	0.5981	0.4713	0.4313	0.1647	0.4369	0.6474	0.2320	0.4163	0.4778
	$ESCM^2$-IPS	0.7080	0.5186	0.8325	0.9326	0.5422	0.6069	0.4801	0.4392	0.1643	0.4423	0.6585	0.2331	0.4232	0.4862
	DCMT	0.6958	0.5079	0.8205	0.9256	0.5301	0.5950	0.4679	0.4305	0.1617	0.4332	0.6454	0.2291	0.4131	0.4768
	CL4CVR	0.7034	0.5054	0.8327	0.9304	0.5381	0.6026	0.4766	0.4356	0.1624	0.4450	0.6564	0.2295	0.4227	0.4832
	DDPO	0.7054	0.5144	0.8371	0.9304	0.5407	0.6064	0.4787	0.4387	0.1653	0.4419	0.6575	0.2333	0.4226	0.4858
	NISE	0.6835	0.5043	0.8285	0.9272	0.5362	0.5969	0.4728	0.4287	0.1644	0.4413	0.6471	0.2306	0.4188	0.4756
	EVI	0.6931	0.5049	0.8182	0.9247	0.5289	0.5936	0.4694	0.4309	0.1590	0.4305	0.6456	0.2257	0.4128	0.4770
	PARS (Ours)	**0.9398**	**0.6173**	0.8530	0.9204	**0.5903**	0.6285	**0.4868**	0.4170	**0.2278**	**0.4761**	0.6393	**0.3066**	0.4395	0.4645
RetailRocket	DeepFM	0.5226	0.3122	0.7048	0.8626	0.4875	0.5666	0.2955	0.2373	0.2126	0.6110	0.7979	0.2381	0.3745	0.3454
	ESMM	0.5206	0.3254	0.7188	0.8690	0.5013	0.5777	0.2990	0.2382	0.2285	0.6220	0.8008	0.2536	0.3802	0.3468
	MMOE	0.5192	0.3177	0.7091	0.8694	0.4916	0.5708	0.2949	0.2372	0.2236	0.6120	0.7992	0.2478	0.3743	0.3455
	BERT4Rec	0.5103	0.3156	0.7244	0.8664	0.4996	0.5732	0.2994	0.2363	0.2198	0.6289	0.8011	0.2452	0.3828	0.3454
	ESM^2	0.5578	0.3420	0.7252	0.8707	0.5274	0.6007	0.2989	0.2375	0.2411	0.6286	0.8044	0.2670	0.3814	0.3466
	$ESCM^2$-DR	0.5210	0.3224	0.7248	0.8754	0.5009	0.5771	0.3006	0.2393	0.2229	0.6263	0.8059	0.2489	0.3822	0.3485
	$ESCM^2$-IPS	0.5261	0.3288	0.7069	0.8588	0.4957	0.5725	0.2970	0.2369	0.2306	0.6095	0.7902	0.2563	0.3752	0.3440
	DCMT	0.5290	0.3173	0.7183	0.8664	0.4997	0.5749	0.2997	0.2384	0.2185	0.6225	0.7997	0.2440	0.3807	0.3469
	CL4CVR	0.5277	0.3020	0.7133	0.8639	0.4905	0.5667	0.2965	0.2369	0.2084	0.6202	0.7987	0.2325	0.3775	0.3451
	DDPO	0.5544	0.3641	0.7384	0.8720	0.5301	0.6000	0.3077	0.2409	0.2609	0.6412	0.8060	0.2875	0.3911	0.3500
	NISE	0.5333	0.3335	0.7171	0.8639	0.5015	0.5778	0.2983	0.2374	0.2331	0.6211	0.8002	0.2595	0.3790	0.3457
	EVI	0.5178	0.3114	0.7074	0.8524	0.4893	0.5636	0.2968	0.2347	0.2132	0.6115	0.7874	0.2386	0.3754	0.3413
	PARS (Ours)	**0.9718**	**0.7835**	**0.9145**	**0.9507**	**0.8076**	**0.8379**	**0.3902**	**0.2687**	**0.6194**	**0.8286**	**0.9030**	**0.6647**	**0.5016**	**0.3914**
Taobao	DeepFM	0.5481	0.1422	0.3162	0.4424	0.2122	0.2669	0.1158	0.1029	0.1102	0.2656	0.3913	0.1197	0.1554	0.1578
	ESMM	0.6808	0.2178	0.4539	0.5575	0.3032	0.3371	0.1663	0.1295	0.1708	0.3860	0.4964	0.1847	0.2243	0.1994
	MMOE	0.6842	0.1886	0.4120	0.5233	0.2692	0.3072	0.1496	0.1212	0.1470	0.3470	0.4634	0.1593	0.2015	0.1863
	BERT4Rec	0.5039	0.1020	0.2767	0.4168	0.1758	0.2350	0.1004	0.0964	0.0783	0.2303	0.3670	0.0853	0.1347	0.1480
	ESM^2	0.7134	0.2296	0.4450	0.5420	0.3002	0.3314	0.1635	0.1264	0.1774	0.3752	0.4800	0.1928	0.2193	0.1939
	$ESCM^2$-DR	0.6996	0.2067	0.4290	0.5390	0.2838	0.3224	0.1570	0.1260	0.1590	0.3607	0.4788	0.1728	0.2108	0.1934
	$ESCM^2$-IPS	0.7099	0.2026	0.4235	0.5341	0.2809	0.3183	0.1554	0.1243	0.1580	0.3570	0.4733	0.1711	0.2085	0.1909
	DCMT	0.6996	0.1978	0.4217	0.5308	0.2776	0.3148	0.1542	0.1239	0.1534	0.3539	0.4701	0.1665	0.2069	0.1900
	CL4CVR	0.6924	0.1930	0.4174	0.5300	0.2739	0.3121	0.1529	0.1231	0.1488	0.3517	0.4686	0.1617	0.2053	0.1891
	DDPO	0.7173	**0.2391**	**0.4690**	**0.5685**	**0.3173**	**0.3491**	**0.1728**	**0.1335**	**0.1875**	**0.3992**	**0.5079**	**0.2026**	**0.2324**	**0.2049**
	NISE	0.6646	0.1945	0.4232	0.5339	0.2762	0.3145	0.1548	0.1246	0.1501	0.3544	0.4734	0.1632	0.2075	0.1911
	EVI	0.6942	0.2082	0.4297	0.5399	0.2858	0.3234	0.1568	0.1259	0.1628	0.3635	0.4795	0.1762	0.2114	0.1934
	PARS (Ours)	**0.8953**	0.1768	0.3476	0.4572	0.2444	0.2926	0.1282	0.1070	0.1407	0.2961	0.4069	0.1514	0.1726	0.1642

demonstrates strong performance across different datasets. On the RetailRocket dataset, PARS outperforms all baselines trained with 100% acquisition labels. While PARS shows lower performance than fully-supervised baselines on the Taobao dataset, it notably surpasses all baselines trained with 50% labels or fewer. On the YooChoose dataset, our method achieves superior results specifically in AUC, HR@1, NDCG@1, Precision@1, Recall@1, and F1@1 metrics compared to baselines using complete label information.

These results highlight the robustness of our approach in label-scarce environments and demonstrate its potential for practical deployment where lack of acquisition labels.

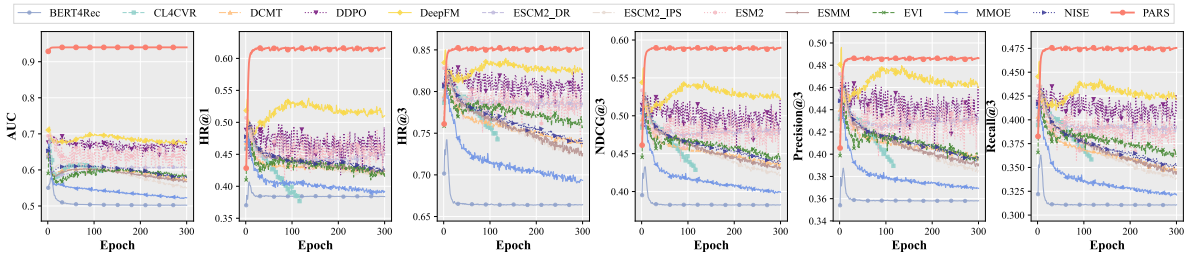

Figure 3.15: Comparison of our method and baselines on the YooChoose dataset across six metrics (AUC, HR@1, HR@3, NDCG@3, Precision@3, Recall@3) over training epochs.

3.4.5.2 Training Stability and Convergence Analysis

To evaluate the training dynamics and convergence behaviour of our proposed methods, we conduct extensive experiments tracking model performance throughout the training process. Figure 3.15 presents a comprehensive comparison between our approach and baseline methods on the YooChoose dataset, evaluating six key metrics: AUC, HR@1, HR@3, NDCG@3, Precision@3, and Recall@3 over 300 training epochs. The results reveal several important insights about our label-free learning framework. First, PARS demonstrates remarkable training stability across all metrics and datasets. While baseline methods exhibit considerable fluctuations during training, particularly in the initial 50 epochs, PARS maintains a smooth and consistent improvement trajectory. This stability is particularly pronounced in ranking metrics such as NDCG@3 and HR@3, where PARS shows monotonic improvement without the oscillations commonly observed in baselines. Second, our approach achieves faster convergence compared to most baselines. On the YooChoose dataset, PARS reaches 95% of its final performance within 50 epochs, while baselines such as DDPO and DeepFM require approximately 150-200 epochs to stabilise. This efficient convergence is attributed to our principled learning framework that leverages inherent data structures rather than relying on potentially noisy label signals. Furthermore, the consistent performance across all six evaluation metrics demonstrates the robustness of PARS. Unlike some baselines that show trade-

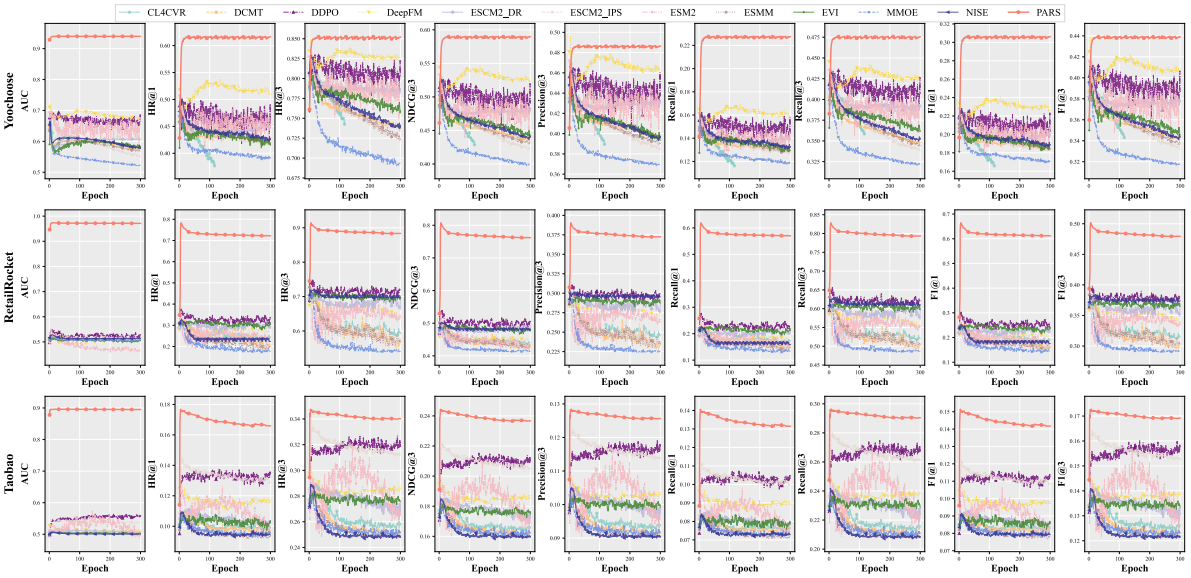

Figure 3.16: Comparison of our method and baselines on the YooChoose, RetailRocket, and Taobao datasets across nine metrics (AUC, HR@1, HR@3, NDCG@3, Precision@3, Recall@1, Recall@3, F1@1, F3@3) over 300 training epochs.

offs between different metrics (e.g., improving AUC while degrading Precision@3), PARS maintains balanced performance improvements across both ranking and classification objectives. This comprehensive superiority validates the effectiveness of PARS.

We have done more experiments to study the stability and convergence further. Figure 3.16 presents the performance metrics of our PARS algorithm compared to baseline methods using 50% of acquisition labels across three datasets over 300 training epochs. Our results demonstrate that PARS achieves rapid convergence while maintaining stable performance with minimal fluctuation throughout the training process. Figure 3.17 provides a detailed view of the first 100 epochs, illustrating the convergence behaviour of PARS against baseline approaches across all three datasets. Notably, PARS converges within the first 25 epochs and maintains stability thereafter, whereas baseline methods exhibit significant oscillations and unstable convergence patterns. This superior convergence behaviour can be attributed to our adaptive selection strategy, which effectively balances exploration and exploitation during the active learning process.

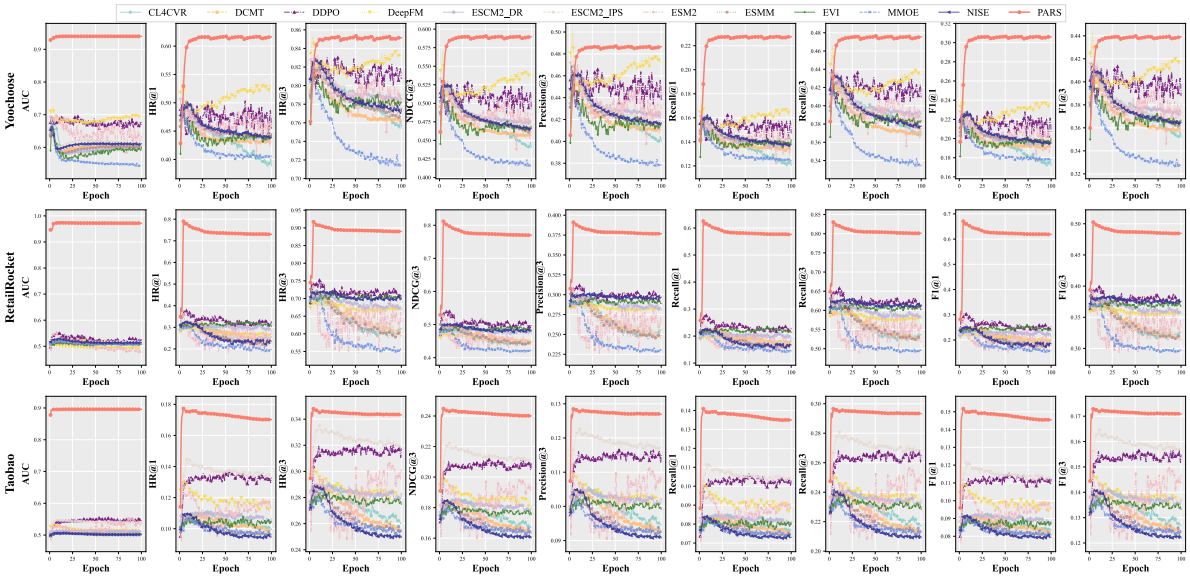

Figure 3.17: Comparison of our method and baselines on the YooChoose, RetailRocket, and Taobao datasets across six metrics (AUC, HR@1, HR@3, NDCG@3, Precision@3, Recall@1, Recall@3, F1@1, F3@3) over 100 training epochs.

The stability of PARS is particularly evident in the reduced variance of performance metrics post-convergence, suggesting that our method is more robust to the inherent stochasticity in the training process. This characteristic is crucial for practical deployments where consistent and predictable model behaviour is essential.

3.5 Discussions

3.5.1 Challenges and Design Rationale

Unlike fully-supervised approaches, our PARS framework operates under a more challenging setting where acquisition labels are not utilised during the training phase. This constraint necessitates learning user conversion patterns solely from historical interaction sequences, making effective representation learning crucial for capturing the latent relationship between users and items.

3.5.2 Key Design Innovations

To address these challenges, we introduce two complementary feature learning mechanisms that enable robust user and item representations:

Self-Supervised Representation Learning. We employ a masked item prediction strategy that randomly masks a proportion of items in user interaction sequences during training. This self-supervised objective encourages the model to learn contextual dependencies and capture meaningful item representations without relying on explicit acquisition labels. The masking strategy serves as an implicit regularisation mechanism, preventing overfitting to spurious patterns in the interaction data.

Hierarchical Feature Aggregation. We design a pooling-based feature fusion mechanism that aggregates item features across the entire user interaction sequence. This approach enables the model to capture both local and global patterns in user behaviour, embedding rich sequential information into the user representation. The hierarchical aggregation preserves temporal dynamics while maintaining computational efficiency.

3.5.3 Integration with Partial Label Learning

The learned representations serve as the foundation for our partial label learning framework, which progressively identifies potential acquisition items from the user's interaction history. This iterative refinement process leverages the uncertainty inherent in partial labels to guide exploration, resulting in more accurate predictions over time.

3.5.4 Empirical Validation

We validate our approach on three real-world e-commerce datasets, demonstrating consistent improvements over existing baselines. The experimental results confirm that our dual representation learning strategy effectively compensates for the absence of

acquisition labels during training, achieving comparable or superior performance to methods with full supervision.

3.5.5 Broader Implications

Our work highlights the potential of combining self-supervised learning with partial label learning for scenarios where complete annotation is prohibitively expensive. The success of PARS suggests that carefully designed representation learning can bridge the gap between limited supervision and model performance, opening new avenues for practical deployment in resource-constrained settings.

3.6 Summary

The Chapter presents a novel and promising RS setting, where only user browsing histories are available without any item acquisition labels. An effective algorithm termed as partial acquisition recommender system (PARS) has been proposed. The key idea is to use maksed language modeling and partial label learning to help extract reliable user and item embeddings to predict item acquisition probabilities. Experiments show that PARS without using item acquisition labels outperforms baselines using item acquisition labels. In the future, we will extend the work in the following aspects. First, build a theoretical foundation of the proposed method. Second, how to effectively incorporate a small amount of item acquisition labels when available.

CHAPTER 4

ROBUST RECOMMENDER SYSTEMS WITH RATING FLIP NOISE

Recommender systems have become important tools in the daily life of human beings since they are powerful in addressing information overload, and discovering relevant and useful items for users. The success of recommender systems largely relies on the interaction history between users and items, which is expected to accurately reflect the preferences of users on items. However, the expectation is easily broken in practice, due to the corruptions made in the interaction history, resulting in unreliable and untrusted recommender systems. Previous works either ignore this issue (assume that the interaction history is precise) or are limited to handling additive noise. Motivated by this, in this Chapter, we study rating flip noise which is widely existed in the interaction history of recommender systems and combat it by modelling the noise generation process. Specifically, the rating flip noise allows a rating to be flipped to any other ratings within the given rating set, which reflects various real-world situations of rating corruption, *e.g.*, a user may randomly click a rating from the rating set and then submit it. The noise generation process is modelled by the noise transition matrix that denotes the

probabilities of a clean rating flip into a noisy rating. A statistically consistent algorithm is afterwards applied with the estimated transition matrix to learn a robust recommender system against rating flip noise. Comprehensive experiments on multiple benchmarks confirm the superiority of our method.

4.1 Introduction

Recommender systems can cope with the information overload problem effectively and provide personalised services for their users Lu et al. (2015). The research of recommender systems can be dated back to three decades ago Goldberg et al. (1992). Since then, we have witnessed significant progress in developing advanced recommendation algorithms. They have been applied to a wide range of areas, including recommendations in e-commerce websites (Lu et al., 2020), recommendations in intelligent web systems (Yao et al., 2013), recommendations in social networks (Fan et al., 2019), and so on. The recommender system is one of the most ubiquitous user-centred artificial intelligence applications in modern information systems (Nguyen et al., 2013).

The personalised recommendation by recommender systems is based on the interaction history between users and items (Lu, 2004; Ye and Lu, 2024, 2023). For example, if in the history of interaction, a user preferred an item with a large rating, recommender systems would recommend items similar to the previously preferred items to the user. Also, if two users have similar profiles, a recommender system would recommend the items preferred by one user to another one. Obviously, the interaction history between users and items plays an important role in achieving effective recommender systems (Zhang et al., 2021a; Anelli et al., 2023).

The vast majority of existing works on recommender systems assume that the interaction history is reliable and accurate (Koren et al., 2021; Sun et al., 2019b). That is to say, the interaction history should precisely reflect user preferences for items. Unfortu-

nately, the assumption is argued to be strong and difficult to hold in many real-world applications, *i.e.*, the interaction history could be untrusted. We detail the issue through the following three situations. First, it is likely to exist *unreliable users* in the stage of scoring items. For example, a user may quickly and randomly click a rating to save time. Second, an untrusted interaction history may come from *negative users* in the stage of scoring items. For example, one user may deliberately provide a score that is contrary to the true personal preference. Third, if recommender systems are attacked by third parties, the ratings of users on items can be changed. When the interaction history is not reliable, recommender systems based on it will be not reliable accordingly, which is pessimistic. In addition, a few previous works design heuristic algorithms to cope with the untrusted interaction history and make robust recommender systems. However, these methods target rating corruptions caused by additive noise (Bishop and Nasrabadi, 2006). Specifically, let the rating set be $\{1,2,3,4,5\}$, *i.e.*, users can only choose ratings from $\{1,2,3,4,5\}$. The additive noise often changes the clean ratings slightly, *e.g.*, from 1 to 2 or from 5 to 4, which is limited in practical tasks.

In this Chapter, to tackle the untrusted interaction history caused by a wide range of rating corruptions, we formally define rating flip noise for recommender systems and propose to construct robust recommender systems via modelling the rating flip noise. Specifically, for a rating provided by a user on an item, the rating flipped noise makes it be flipped to any other ratings in the rating set, which is hence more realistic than prior rating corruptions via additive noise. The rating flip noise allows the occurrence of extreme hazards in practice, *e.g.*, from the highest (*resp.*, the lowest) rating to the lowest (*resp.*, the highest) rating, which is more challenging. To tackle rating flip noise, we propose to employ the noise transition matrix (Patrini et al., 2017) to model it, which builds the relationship between clean ratings and flipped ratings. Intuitively, given the relationship between clean ratings and flipped ratings and the noisy ratings, clean rating

information can be inferred. Robust recommender systems can therefore be designed.

Technically, we transform the recommendation problem into a multi-class classification problem, where the number of classes is equal to the number of optional ratings. Then, the label noise transition matrix studied in the community of learning with noisy labels (Yao et al., 2020) can be employed to model the rating flip noise. Afterwards, to estimate the noise transition matrix, we build an unbiased estimator by identifying anchor points (Liu and Tao, 2015) belonging to different classes (ratings). A statistically consistent algorithm (Xia et al., 2019) is lastly applied to learn a robust recommend system against rating flip noise.

Contributions. Before delving into details, we highlight our contributions as follows:

- We study the problem of how to handle a new type of noise, i.e., rating flip noise, for building robust recommender systems, which is novel and significant for the community of recommender systems.
- We introduce a new but more practical noise regarding the ratings in recommend systems, named rating flip noise. There are many real-world situations that contain this noise as discussed in the Introduction.
- To combat rating flip noise, we propose to utilize the noise transition matrix to model it and build an unbiased estimator for the learning of the transition matrix.
- We conduct extensive experiments on the datasets corrupted by rating flipped noise. The results demonstrate the superiority of our algorithm over existing state-of-the-art methods. In addition, detailed analyses on the results are provided.

Organization. The rest of this Chapter is organised as follows. In Section 4.2, we introduce background knowledge and review related literature in both recommendation models and learning with noisy labels. In Section 4.3, we discuss the technical details of the proposed method for modelling and handling rating flip noise. In Section 4.4, we

conduct a series of experiments to show that our method can improve the robustness of a wide range of recommendation models. Lastly, we conclude this Chapter in Section 4.5.

4.2 Background

In this section, we will briefly introduce the related work beyond recommendation models, i.e., discussing the research progresses in robust recommender systems and learning with noisy labels.

4.2.1 Research Progresses in Robust Recommendation Models

Towards building more reliable recommender systems that better adapt to realistic scenes, there are a series of methods working in improving the robustness of recommendation models, which can be divided into two groups, *i.e.*, supervised classification methods and unsupervised clustering methods. In more detail, supervised classification methods begin with feature engineering and then move on to algorithm development (Ge et al., 2022). For example, Yang et al. (2016) extracts well-designed features from user profiles to identify attack profiles. (Yuan et al., 2023a) develop an innovative defense strategy against poisoning attacks, which involves the implementation of hierarchical gradient clipping with sparsified updating, specifically designed to counteract prevailing poisoning attack techniques. Yuan et al. (2023b) proposes an efficient unlearning method FRU (Federated Recommendation Unlearning), inspired by the log-based rollback mechanism of transactions in database management systems, to enhance robustness. Yuan et al. (2023c) proposes a new kind of poisoning attack for visually-aware federated recommender systems, namely image poisoning attacks, where adversaries can gradually modify the uploaded image to manipulate item ranks during federated recommender systems' training process. It also employs a diffusion model to purify each uploaded image and detect the adversarial images. Wang et al. (2021); Xu et al. (2021b); Yu et al.

(2023) propose to employ a deconfounded mechanism to alleviate the bias in the recommendation to further improve the robustness of recommender systems. Unsupervised clustering methods aim to group individuals into groups and then eliminate suspicious ones (Ge et al., 2022). For instance, Zhang et al. (2015) proposes a propagation method based on matrix interaction to calculate the likelihood of each user being an attacked user based on a small set of trusted users. Additionally, Zhang et al. (2017) design a robust collaborative filtering method by incorporating non-negative matrix factorisation with R1-norm. We refer readers to the surveys (Ge et al., 2022; Wang et al., 2022b) for more advanced works in robust recommender systems.

4.2.2 Research Progresses in Learning with Noisy Labels

Learning with noisy labels is one of the hottest topics in weakly supervised learning (Sugiyama et al., 2022), which is closely related to this work. Methods for dealing with noisy labels can be divided into two groups: model-free and model-based algorithms. In the first group, a series of methods handle noisy labels without modelling label noise, *e.g.*, extracting confident examples with small losses for robust training (Jiang et al., 2018; Han et al., 2018), designing robust loss functions (Ghosh et al., 2015; Zhou et al., 2021), introducing implicit/explicit regularization (Fatras et al., 2021; Wei et al., 2021), and etc. Although these algorithms empirically work well, without modelling the label noise explicitly, their reliability cannot be guaranteed.

This motivates researchers to model and learn label noise (Liu and Tao, 2015). The noise transition matrix $\boldsymbol{T}$ (Patrini et al., 2017) was proposed to explicitly model the generation process of label noise. Mathematically, we have $T_{ij}(\boldsymbol{x}) = P(\tilde{Y} = j|Y = i, X = \boldsymbol{x})$, where X denotes the random variable for the instance, $\tilde{Y}$ for the noisy label, and Y for the latent clean label. If the noise transition matrix is learned precisely, statistically consistent algorithms, *e.g.*, classifier-consistent (Patrini et al., 2017) and risk-consistent

ones (Liu and Tao, 2015), can be applied to train robust models. Based on this, previous works propose multiple methods to learn the noise transition matrix (Goldberger and Ben-Reuven, 2017; Yao et al., 2020; Jiang et al., 2022).

Unfortunately, at the present stage, the learning of the noise transition matrix is limited in image or text classification (Song et al., 2022). There is no attempt about learning and utilise the noise transition matrix to deal with the preference noise in recommendations. Therefore, this Chapter makes this attempt and discusses how to learn the noise transition matrix for tackling noisy preferences in recommender systems.

4.3 Modelling Rating Flip Noise via Noise Transition Matrix

In this section, firstly, we will demonstrate preliminaries by introducing the problem setup and the rating flip noise; secondly, how to model the rating flip noise will be discussed; Finally, we will present our new method to build robust recommender systems to deal with the flip rating noise.

4.3.1 Preliminaries

Problem setup. Suppose that there are m users and n items, with $\mathbf{Y} \in R^{m \times n}$ is a user-item interaction behavior matrix. The element y_{ij} of the interaction matrix denotes the rating of the user i to the item j. The rating set is $\{1,2,\ldots,K\}$. In other words, all ratings are chosen from $\{1,2,\ldots,K\}$. Due to the corruption on ratings by noise, we can only observe a noisy interaction matrix denoted by $\tilde{\mathbf{Y}} \in R^{m \times n}$. Without accessing the clean rating in $\mathbf{Y}$, by employing $\tilde{\mathbf{Y}}$, our aim is to learn a robust recommendation model that can produce the ratings close to clean ones. Note that noisy ratings do not mean that they are incorrect. Some of them are correct and some of them are incorrect. But we

do not know which are which.

Rating flip noise. We propose rating flip noise to simulate a popular corruption process of ratings in practice. Specifically, for a clean rating, rating flip noise could pollute it by modifying this rating to any other ratings. For example, the set of ratings is $\{1,2,3,4,5\}$. If the clean rating is 5, rating flip noise makes it be flipped to any other ratings. Different from previous work in robust recommendations (Ge et al., 2022), which mostly allows the rating flip to nearby ratings (*e.g.*, 1 to 2, 4 to 5), rating flip noise is more realistic and more challenging. In practice, it is likely that a user quickly and randomly clicks ratings to save time. In this case, original ratings can be flipped into any other ratings. As the rating flip noise makes the extreme flipping happen (*e.g.*, $1 \rightarrow 5$ and $5 \rightarrow 1$), existing methods working in robust recommendations cannot well address it. We provide empirical evidence in Section 4.4. Below we discuss the technical details of modelling rating flip noise and enhance the robustness of recommender systems with the modelling.

4.3.2 Modelling Rating flip Noise

To gain a comprehensive understanding of our proposed method, let's discuss its overall model structure at a high level. Our method aims to model rating flip noise, and to do so, we begin by learning representations of both users and items. These representations are pivotal as they enable the model to effectively capture and process user and item information. Next, we recognise that ratings are essentially the scores provided by users for specific items. To harness this information, we create instances by combining the representations of users and items. These instances then form the basis for constructing instance-label pairs. With the instance-label pairs in place, we can proceed to learn the mapping from these instances to their corresponding labels. This mapping is crucial in understanding and mitigating the effects of rating flip noise. Additionally, it allows us to

obtain the posteriors of anchor points, which are vital for estimating the noise transition matrix accurately. In the following, we will delve into the details of our procedure, providing a step-by-step breakdown of our methodology. This will help clarify the logical flow of our method and how each component contributes to modelling and addressing rating flip noise effectively.

Representation learning. The objective is to learn a user embedding matrix and an item embedding matrix, with $\mathbf{p}_u \in R^{1\times d}$ and $\mathbf{q}_v \in R^{1\times d}$ denote the representation parameters for the user u and the item v, respectively, where d represent the dimension of the representations. Based on various methods that work with different types of input data, we can obtain the representations of users and items conveniently, *e.g.*, with denoising auto-encoders (Wu et al., 2016) and variational auto-encoders (Liang et al., 2018). Note that during representation learning, we only have a noisy user-item interaction behaviour matrix $\tilde{\mathbf{Y}}$. The early-stopping technology is applied to reduce the side-effect of the noisy matrix on representations, which has been widely used to avoid the model from overfitting noise (Li et al., 2020; Bai et al., 2021b). After representation learning, we can achieve the representations of all users, *i.e.*, $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_u, \ldots, \mathbf{p}_m]$, where m represents the number of users, and the representations of all items, *i.e.*, $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_v, \ldots, \mathbf{q}_n]$, where n represents the number of items.

Instance-label pair construction. We first construct a training instance with the learned representations. In this work, we build an instance by using a user-item pair. In more details, given a user-item pair $(\mathbf{p}_u, \mathbf{q}_v)$, we concatenate the two vectors horizontally to obtain an instance $\mathbf{x}_{uv} = [\mathbf{p}_u, \mathbf{q}_v] \in R^{1\times 2d}$. By further employing the noisy interaction matrix $\tilde{\mathbf{Y}}$, we can construct uv instance-label pairs denoted by $\{(\mathbf{x}_i, \tilde{y}_i)\}_{i=1}^{mn}$. As the value of the rating is discrete, the pairs can be regarded as noisy training data for a classification task, where the noisy label $\tilde{y}$ is in $\{1, \ldots, K\}$.

Transition matrix for noise modeling. The noise transition matrix $\boldsymbol{T}$ (Patrini

et al., 2017) was proposed to model the generation process of classification label noise. In this Chapter, we employ it to model the rating flip noise. Generally, the transition matrix depends on instances. However, given only noisy data, the instance-dependent transition matrix is non-identifiable without any additional assumption (Xia et al., 2019). In this Chapter, we therefore study the class-dependent and instance-independent transition matrix for simplicity, *i.e.*, $T_{ij}(\mathbf{x}) = P(\tilde{Y} = j|Y = i, X = \mathbf{x}) = P(\tilde{Y} = j|Y = i)$, which is identifiable under mild conditions. Note that even the real-world rating flip noise is instance-dependent, the class-dependent and instance-independent transition matrix could approximately model it well.

Anchor points. To learn the noise transition matrix, the concept of anchor point was proposed (Scott, 2015; Liu and Tao, 2015). If an instance $\mathbf{x}$ is an anchor point for the class i, $P(Y = i|X = \mathbf{x})$ is equal to one or close to one, where X denotes the variable of the instance, and Y denotes the variable of the clean label. Given an instance, if $P(Y = i|X = \mathbf{x}) = 1$, we have that for $k \neq i$, $P(Y = k|X = \mathbf{x}) = 0$. Then, we have

$$P(\tilde{Y} = j|X = \mathbf{x}) = \sum_{k=1}^{K} T_{kj} P(Y = k|X = \mathbf{x}) = T_{ij}. \tag{4.1}$$

Obviously, the transition matrix $\boldsymbol{T}$ can be estimated via estimating the noisy class posterior probabilities for anchor points (Liu and Tao, 2015; Xia et al., 2019; Yao et al., 2020). Note that without anchor points, the other data points cannot capture the true simplex of the true transition matrix, leading to poor estimation, which is illustrated in Figure 4.1.

To make the outputs of a recommendation model be in a probabilistic form, we follow the work He et al. (2018) and transform the input user and item representations to outputs. A Softmax activation function is employed to normalise the outputs. Taking the instance $\mathbf{x}$ as an input, the model output after the Softmax activation function is denoted by $f(\mathbf{x}) \in R^K$. Based on the probabilistic model outputs, to identify anchor points from

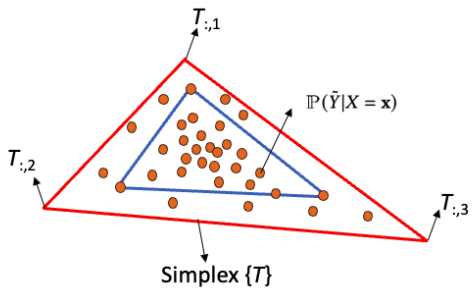

Figure 4.1: Geometric illustration of the problem of estimating the transition matrix without anchor points (3 class classification). The red triangle is the simplex of $\boldsymbol{T}$ with vertices denoted by $\boldsymbol{T}_{:,i}$. When there are no anchor points, the simplex found with existing methods (blue triangle) by using extreme-valued noisy class-posterior probabilities is not the true simplex (red triangle).

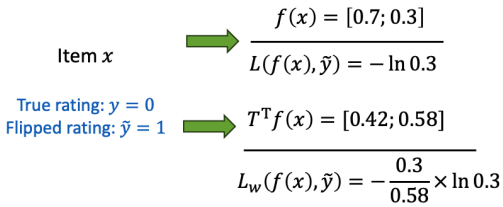

Figure 4.2: The illustration of the importance reweighting for handling noisy ratings. The true noise transition matrix is [0.7, 0.3; 0.3, 0.7], *i.e.*, the noise rate is 30%. The loss L denotes the unweighted loss by importance reweighting. The loss L_w denotes the weighted loss. With the importance reweighting way, the loss of the incorrect rating is reduced effectively, leading to enhanced model robustness.

noisy data for the estimation of the noise transition matrix, we follow (Patrini et al., 2017; Xia et al., 2019; Yao et al., 2020) to locate anchor points belonging to different classes based on noisy class probability estimates. Then the transition matrix for modelling rating flip noise can be learned via Eq. (4.1) and the identified anchor points. Note that if there is no anchor points existing in the training data, we can use the instance with the largest class posterior as an approximation.

4.3.3 Statistically Consistent Algorithms for Robust Recommendations

After estimating the noise transition matrix $\boldsymbol{T}$ to model rating flip noise, we can build statistically consistent algorithms for robustly training of recommendation models. The statistically consistent algorithms can be grouped into classifier-consistent algorithms and risk-consistent algorithms. In more details, if an algorithm is classifier-consistent, by increasing the size of noisy data, the learned model will converge to the optimal model learned by clean data (Patrini et al., 2017). If an algorithm is risk-consistent, by increasing the size of noisy data, the empirical risk calculated by noisy data and a modified loss function will converge to the expected risk calculated by clean data and the original loss function (Liu and Tao, 2015). Here we exploit the risk-consistent algorithm, since prior work (Xia et al., 2019) shows that the risk-consistent algorithm is more robust than the classifier-consistent algorithm when handling noisy labels.

Given the constructed noisy data with rating flip noise $\{(\mathbf{x}_i, \tilde{y})\}_{i=1}^{mn}$, we exploit the importance reweighting technique (Xia et al., 2019). We first explain why the importance reweighting technique in handling rating flip noise is risk-consistent. Specifically, we have

$$
\begin{aligned}
&R(f) = \mathbb{E}_{(X,Y)\sim D}[\ell(f(X),Y)] = \int_{\mathbf{x}} \sum_i P_D(X=\mathbf{x}, Y=i)\ell(f(\mathbf{x}),i)d\mathbf{x} \\
&= \int_{\mathbf{x}} \sum_i P_{\tilde{D}}(X=\mathbf{x}, \tilde{Y}=i)\frac{P_D(X=\mathbf{x}, \tilde{Y}=i)}{P_{\tilde{D}}(X=\mathbf{x}, \tilde{Y}=i)}\ell(f(\mathbf{x}),i)d\mathbf{x} \\
&= \int_{\mathbf{x}} \sum_i P_{\tilde{D}}(X=\mathbf{x}, \tilde{Y}=i)\frac{P_D(\tilde{Y}=i|X=\mathbf{x})}{P_{\tilde{D}}(\tilde{Y}=i|X=\mathbf{x})}\ell(f(\mathbf{x}),i)d\mathbf{x} = \mathbb{E}_{(X,Y)\sim \tilde{D}}[\bar{\ell}(f(X),Y)],
\end{aligned}
$$

where $\ell(\cdot)$ denotes any loss function for classification, D denotes the distribution for clean data, $\tilde{D}$ for noisy data, $\bar{\ell}(f(\mathbf{x}),i) = \frac{P_D(\tilde{Y}=i|X=\mathbf{x})}{P_{\tilde{D}}(\tilde{Y}=i|X=\mathbf{x})}\ell(f(\mathbf{x}),i)$, and the second last equation holds because noise only exists on label instead of instance $\mathbf{x}$. Since $P(\tilde{Y}|X=\mathbf{x}) = T^\top P(Y|X=\mathbf{x})$

and that the diagonal entries of (learned) transition matrices for label-noise learning are all much larger than zero, $P_D(\tilde{Y}=i|X=\mathbf{x})\neq 0$ implies $P_{\tilde{D}}(\tilde{Y}=i|X=\mathbf{x})\neq 0$.

Note that we only have a finite training sample. We hence use the empirical risk to approximate the expected risk. The objective function in optimisation is

$$\hat{R}_{\text{R-C}}(f)=\frac{1}{mn}\sum_{i=1}^{mn}\frac{f_{\tilde{y}_i}(\mathbf{x}_i)}{(\boldsymbol{T}^\top f)_{\tilde{y}_i}(\mathbf{x}_i)}L(f(\mathbf{x}_i),\tilde{y}_i), \tag{4.2}$$

where $L(\cdot)$ denotes the cross entropy loss. For better understanding, we provide the illustration to explain the effectiveness of importance reweighting on one example, which is shown in Figure 4.2. Notice that we detect the anchor points from the noisy training data. Data points that are similar to anchor points will be detected if there are no anchor points available (Ghosh et al., 2015). Besides, recommendation models may have poor confidence calibration (Yao et al., 2020). The issues make an inaccurate estimation of the noise transition matrix. To address the issues, we follow the work (Xia et al., 2019) to revise the estimated transition matrices with a slack variable $\Delta\boldsymbol{T}$, which helps lead to a more robust recommendation model. In more detail, we minimize

$$\hat{R}_{\text{R-C}}(f,\Delta\boldsymbol{T})=\frac{1}{mn}\sum_{i=1}^{mn}\frac{f_{\tilde{y}_i}(\mathbf{x}_i)}{((\hat{\boldsymbol{T}}+\Delta\boldsymbol{T})^\top f)_{\tilde{y}_i}(\mathbf{x}_i)}L(f(\mathbf{x}_i),\tilde{y}_i), \tag{4.3}$$

where $\hat{\boldsymbol{T}}$ is the estimated transition matrix. For the minimisation of Eq. (4.3), we use the output of minimisation with Eq. (4.2) for the initialisation of f. Note that the risk-consistent algorithm can facilitate the revision of the noise transition matrix, since the heuristic for tuning the transition matrix is that a favourable transition matrix would make the classification risk w.r.t. clean data small.

4.4 Experiments

Baselines. We exploit several advanced methods as baselines: (1) Collaborative filtering (CF) (Sarwar et al., 2001); (2) Neural collaborative filtering (NCF) (He et al.,

Table 4.1: Experimental results of evaluations on collaborative filtering recommendations with simulated noisy MovieLens-100k and MovieLens-1M. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy). The best result in each case is shown in **boldface**.

MovieLens-100k						
Metric	Method/Noise	Clean	10%	20%	30%	40%
MAE ↓	CF	0.087	0.171±0.005	**0.223±0.016**	0.332±0.028	0.413±0.035
	NCF	0.088	0.175±0.011	0.242±0.025	0.365±0.041	0.424±0.047
	ONCF	0.092	0.172±0.004	0.238±0.009	0.323±0.017	0.417±0.026
	CML	0.089	0.170±0.004	0.248±0.015	0.352±0.037	0.405±0.041
	HCCF	0.095	0.171±0.003	0.246±0.016	0.337±0.008	0.406±0.023
	NCL	0.098	**0.165±0.003**	0.245±0.009	0.340±0.018	0.425±0.020
	JNCF	0.091	0.174±0.006	0.250±0.014	0.342±0.026	0.430±0.031
	RNCF	0.090	0.170±0.002	0.246±0.017	0.339±0.038	0.407±0.045
	RRFN	0.089	0.174±0.002	0.241±0.005	**0.330±0.020**	**0.400±0.013**

MovieLens-100k						
Metric	Method/Noise	Clean	10%	20%	30%	40%
Accuracy ↑	CF	0.945	0.937±0.007	0.893±0.008	0.815±0.021	0.664±0.027
	NCF	0.945	0.919±0.012	0.895±0.019	0.797±0.031	0.673±0.044
	ONCF	0.928	0.912±0.007	0.882±0.025	0.792±0.030	0.640±0.026
	CML	0.930	0.923±0.004	0.892±0.015	0.802±0.028	0.695±0.048
	HCCF	0.947	**0.943±0.003**	**0.902±0.007**	0.793±0.014	0.666±0.026
	NCL	0.950	0.940±0.004	0.893±0.005	0.801±0.017	0.661±0.024
	JNCF	0.918	0.909±0.005	0.899±0.007	0.782±0.039	0.655±0.042
	RNCF	0.937	0.935±0.002	0.900±0.009	0.807±0.019	0.704±0.043
	RRFN	0.942	0.932±0.004	0.893±0.006	**0.819±0.015**	**0.707±0.040**

MovieLens-1M						
Metric	Method	Clean	10%	20%	30%	40%
MAE ↓	CF	0.089	0.174±0.004	0.258±0.005	0.350±0.012	0.436±0.024
	NCF	0.087	0.180±0.002	0.259±0.003	0.345±0.017	0.425±0.017
	ONCF	0.094	0.202±0.010	0.279±0.015	0.339±0.024	0.433±0.028
	CML	0.093	0.179±0.006	**0.254±0.012**	0.347±0.015	0.415±0.030
	HCCF	0.094	0.180±0.005	0.259±0.018	0.350±0.009	0.420±0.028
	NCL	0.092	0.172±0.008	0.258±0.017	0.345±0.020	0.410±0.031
	JNCF	0.096	0.196±0.014	0.256±0.019	0.352±0.028	0.425±0.033
	RNCF	0.086	**0.165±0.007**	0.264±0.015	0.353±0.022	0.393±0.029
	RRFN	0.088	0.172±0.006	0.258±0.009	**0.328±0.017**	**0.389±0.026**

MovieLens-1M						
Metric	Method	Clean	10%	20%	30%	40%
Accuracy ↑	CF	0.942	0.934±0.002	0.876±0.005	0.792±0.030	0.643±0.018
	NCF	0.945	0.920±0.002	0.876±0.007	0.806±0.012	0.653±0.021
	ONCF	0.935	0.922±0.007	0.902±0.006	0.827±0.019	0.727±0.036
	CML	0.937	0.921±0.003	0.894±0.010	0.811±0.020	0.708±0.029
	HCCF	0.945	**0.940±0.004**	0.900±0.012	0.814±0.015	0.719±0.028
	NCL	0.951	0.936±0.003	0.897±0.006	0.825±0.020	0.721±0.032
	JNCF	0.926	0.917±0.006	0.895±0.013	0.797±0.030	0.711±0.024
	RNCF	0.939	0.929±0.003	0.900±0.003	0.821±0.012	0.735±0.042
	RRFN	0.942	0.929±0.004	**0.906±0.007**	**0.836±0.015**	**0.742±0.038**

2017); (3) Outer neural collaborative filtering (ONCF) (He et al., 2018); (4) Collaborative metric learning (CML) (Hsieh et al., 2017); (5) Hypergraph contrastive collaborative filtering (HCCF) (Xia et al., 2022); (6) Neighborhood-enriched Contrastive collaborative filtering (NCL) (Lin et al., 2022); (7) Joint neural collaborative filtering (JNCF) (Chen et al., 2019); (8) R1-norm neural collaborative filtering (RNCF) (Zhang et al., 2017). Note that baselines (5), (6), (7), and (8) have been demonstrated that they can improve the robustness of recommendation models, which are appreciated in realistic tasks. As our method builds robust recommendations against rating flip noise, we name it as RRFN below. We empirically show the robustness of the proposed methods compared with the above baselines on three datasets, *i.e.*, MovieLens, Jester, and Jester-21.

4.4.1 Robusifying Recommendation on Movie Ratings

4.4.1.1 Experimental Setup

Datasets. We employ MovieLens-100k/1M[1] here. MovieLens-100k is a stable benchmark dataset about movie ratings, which contains 100,000 ratings from 1,000 users on 1,700 movies. MovieLens-1M includes 1 million ratings from 6,000 users on 4,000 movies. As did in (He et al., 2017), we learn implicit feedback. If there is interaction between user and item, then the target value will be 1; otherwise 0. Note that the contextual data of the datasets, *e.g.*, the user age and gender, are not used. As the ratings of original MovieLens-100k/1M are clean, we inject rating flip noise into them. The noise rate ρ is set to 10%, 20%, 30%, and 40% respectively. Namely, there are ratings of a fraction ρ being flipped.

Implementations. For a fair comparison, we implement all methods with default parameters by PyTorch on NVIDIA GTX3090 GPUs. We use an Adam optimizer (Kingma and Ba, 2014), batch size 256, and learning rate 0.001. The parameter of dropout in

[1]https://grouplens.org/datasets/movielens/

Table 4.2: Experimental results of evaluations on collaborative filtering recommendations with simulated noisy Jester. Symmetric rating flip noise is employed. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy). The best result in each case is shown in **boldface**.

Metric	Method	Clean	Sym. 10%	Sym. 20%	Sym. 30%	Sym. 40%
MAE ↓	CF	0.527	0.602±0.026	0.652±0.019	0.700±0.014	0.720±0.020
	NCF	0.522	0.566±0.012	0.592±0.014	0.627±0.027	0.666±0.037
	ONCF	0.513	0.550±0.032	0.580±0.023	0.619±0.034	0.652±0.030
	CML	0.513	0.548±0.003	0.575±0.024	0.609±0.026	0.644±0.029
	HCCF	0.509	0.546±0.020	0.570±0.018	0.605±0.029	0.650±0.024
	NCL	0.508	0.543±0.009	0.562±0.018	0.609±0.012	0.637±0.026
	JNCF	0.529	0.542±0.017	0.560±0.026	0.603±0.019	0.640±0.025
	RNCF	0.521	**0.540±0.023**	0.552±0.034	**0.600±0.036**	0.635±0.032
	RRFN	0.522	0.553±0.022	**0.545±0.026**	**0.600±0.030**	**0.610±0.025**
Metric	Method	Clean	Sym. 10%	Sym. 20%	Sym. 30%	Sym. 40%
Accuracy ↑	CF	0.546	0.449±0.028	0.420±0.018	0.350±0.024	0.286±0.023
	NCF	0.541	0.514±0.015	0.500±0.012	0.465±0.028	0.417±0.025
	ONCF	0.554	0.520±0.020	0.513±0.018	0.477±0.026	0.431±0.029
	CML	0.550	0.525±0.017	0.515±0.026	0.482±0.026	0.435±0.034
	HCCF	0.563	0.528±0.006	0.512±0.017	0.482±0.021	0.446±0.018
	NCL	0.569	0.529±0.003	0.525±0.010	0.474±0.025	0.440±0.022
	JNCF	0.548	0.530±0.015	0.522±0.037	0.493±0.021	0.440±0.029
	RNCF	0.550	0.532±0.034	**0.527±0.036**	0.501±0.014	0.458±0.034
	RRFN	0.551	**0.535±0.006**	0.520±0.018	**0.508±0.019**	**0.466±0.026**

learning user and item presentations is set to 0.2. The dimension of presentations is set to 32 by default. We train recommendation models 10 epochs in total. For the representation learning of users and items, we inherit the method ONCF. For evaluation protocols, we employ two measurement metrics that are popularly used in recommender systems. That is to say, we utilise the mean absolute error and the rating prediction accuracy. For each simulated case, we perform five independent trials with different random seeds. The mean and standard deviation of experimental results are reported.

4.4.1.2 Comparisons with Baseline Methods

We provide experimental results on simulated noisy MovieLens-100k and MovieLens-1M in Table 4.1. First, we can observe that, as training data becomes increasingly noisy (*i.e.*,

Table 4.3: Experimental results of evaluations on collaborative filtering recommendations with simulated noisy Jester. Pairflip rating flip noise is exploited. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy). The best result in each case is shown in **boldface**.

Metric	Method	Clean	Pair. 10%	Pair. 20%	Pair. 30%	Pair. 40%
MAE ↓	CF	0.527	0.617±0.006	0.662±0.029	0.702±0.013	0.729±0.018
	NCF	0.522	0.560±0.015	0.587±0.025	0.612±0.032	0.640±0.033
	ONCF	0.513	0.548±0.013	0.568±0.026	0.612±0.030	0.638±0.037
	CML	0.513	0.552±0.016	0.562±0.027	0.606±0.027	0.650±0.041
	HCCF	0.509	**0.530±0.007**	0.560±0.009	0.609±0.012	0.646±0.026
	NCL	0.508	0.534±0.010	0.555±0.023	0.602±0.019	0.639±0.021
	JNCF	0.529	0.545±0.012	0.560±0.032	0.605±0.019	0.662±0.026
	RNCF	0.521	0.535±0.006	**0.553±0.015**	0.594±0.022	0.630±0.029
	RRFN	0.522	0.572±0.015	0.579±0.020	**0.589±0.016**	**0.596±0.019**
Metric	Method	Clean	Pair. 10%	Pair. 20%	Pair. 30%	Pair. 40%
Accuracy ↑	CF	0.546	0.450±0.015	0.415±0.024	0.353±0.028	0.282±0.017
	NCF	0.541	0.518±0.017	0.494±0.020	0.467±0.029	0.423±0.047
	ONCF	0.554	0.527±0.006	0.507±0.015	0.473±0.024	0.440±0.032
	CML	0.550	0.529±0.012	0.512±0.020	0.478±0.027	0.412±0.043
	HCCF	0.563	0.530±0.012	0.504±0.016	0.460±0.017	0.442±0.028
	NCL	0.569	**0.540±0.007**	0.510±0.012	0.473±0.012	0.467±0.025
	JNCF	0.548	0.532±0.014	0.508±0.023	0.473±0.018	0.408±0.017
	RNCF	0.550	0.519±0.028	0.506±0.033	0.470±0.038	0.445±0.026
	RRFN	0.551	0.524±0.017	**0.522±0.021**	**0.513±0.024**	**0.504±0.027**

higher noise rates), the performance of all methods becomes worse. Namely, larger MAEs and lower accuracies will be achieved. Second, we can see that when the noise level is low, *i.e.*, 10% and 20%, our method can only sometimes achieve the best result (*e.g.*, the accuracy of MovieLens-1M with 20% noise). In other cases, although our method cannot work best, the achieved performance is competitive. When the noise level increase, *i.e.*, 30% and 40%, the classification task becomes more challenging. Our method always enjoys the best results with large margins. Overall, the experiments on simulated noisy MovieLens-100k and MovieLens-1M well demonstrate the effectiveness of the proposed method against rating flip noise.

4.4.2 Robusifying Recommendation on Joker Ratings

4.4.2.1 Experimental Setup

Datasets. We exploit Jester[2], where data is about anonymous ratings of jokes by users of the Jester Joke Recommender System. The ratings in the original Jester are continuous in the range of [-10:10]. We first discretise them for follow-up tasks, where the minimum and maximum values are -10 and 10, respectively. Afterwards, we integrate the discrete ratings into super-ratings. For a rating w, we integrate as $-10 \leq w < -5 \rightarrow w = 1$, $-5 \leq w < 0 \rightarrow w = 2$, $0 \leq w < 5 \rightarrow w = 3$, and $5 \leq w \leq 10 \rightarrow w = 4$. We divide into the whole Jester data into training data and test data, where the ratio is 99:1. We inject two kinds of rating flip noise into the original Jester dataset for the simulation of noisy training data. Specifically, we employ symmetric noise (abbreviated as Sym.) (Patrini et al., 2017) and pairflip noise (abbreviated as Pair.) (Han et al., 2018). The noise rate is set to 10%, 20%, 30%, and 40%, respectively. Note that the pairflip noise is observed to be highly harmful, especially when the noise rate is large, *i.e.*, 30% and 40%. We provide the illustrations of the rating flip noise process about symmetric and pairflip noise in Figure 4.3.

Note that for symmetric noise, if there are four ratings (1, 2, 3, 4) in total and the noise rating is 40%, for instance with the rating 1, the ratings of 40% instances will be flipped into the ratings 2, 3, and 4 uniformly. In this case, the rating 1 can be flipped into the ratings 3 and 4. When the sample size is large, the flip will occur certainly. For pairflip noise, all the ratings are flipped to nearby ratings. Still, consider four ratings in total, the flip will be $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 4$, and $4 \rightarrow 1$. As can be seen, the process $4 \rightarrow 1$ generates extreme noise, which is challenging. If we normally treat a rating larger than 2 as positive and negative otherwise, combining the above discussions, the rating 4 is originally regarded to be positive. After being flipped to be 1, it will be regarded to be

[2]https://eigentaste.berkeley.edu/dataset/

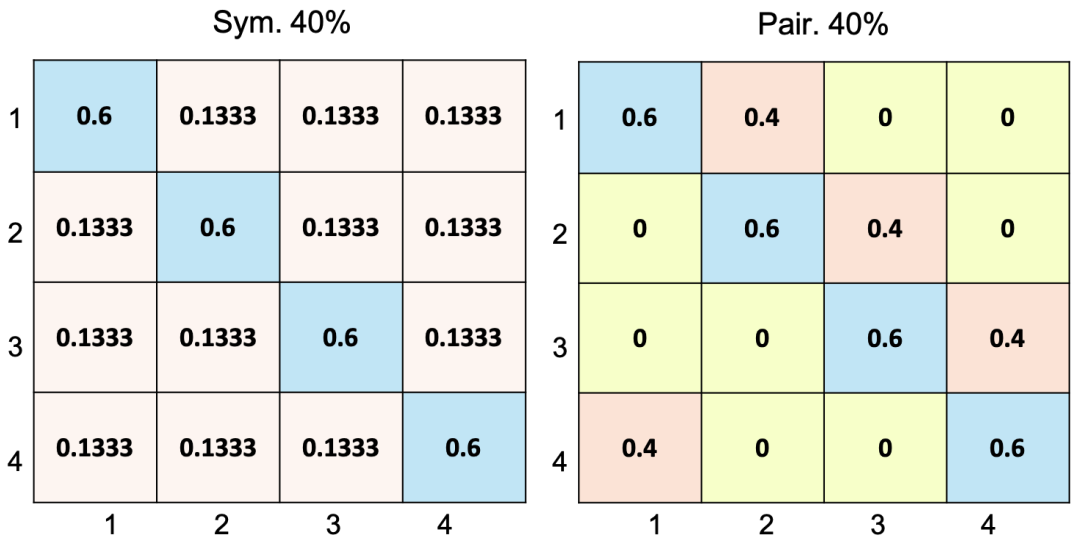

Figure 4.3: The illustrations of the rating flip noise process, where four types of ratings are considered. Left: the rating flip noise process about symmetric noise with a 40% noise rate. Right: the rating flip noise process about pairflip noise with a 40% noise rate.

negative.

Implementation. We use an Adam optimizer (Kingma and Ba, 2014), batch size 256, and learning rate 0.001. The parameter of dropout in learning user and item presentations is set to 0.5. The dimension of presentations is set to 64 by default. We train recommendation models 5 epochs in total. For evaluation metrics, we use MAE and Accuracy as the experiments on MovieLens-100k and MovieLens-1M. In addition, for each simulated case, we perform five independent trials with different random seeds. The mean and standard deviation of experimental results are reported.

4.4.2.2 Comparisons with Baseline Methods

We present the experimental results on simulated noisy Jester in Table 4.2 and Table 4.3. Keep the same as the experiments on MovieLens, with the increase of the noise rate, we can see that the classification performance of different methods on clean test data is deteriorating, which illustrates the side effect of rating flip noise. As for symmetric and

Table 4.4: Experimental results of evaluations on collaborative filtering recommendations with simulated noisy Jester-21. Symmetric rating flip noise is exploited. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy). The best result in each case is shown in **boldface**.

Metric	Method	Clean	Sym. 20%	Sym. 40%	Sym. 60%	Sym. 80%
MAE ↓	CF	0.865	0.881±0.003	0.902±0.004	0.932±0.004	0.937±0.008
	NCF	0.852	0.880±0.005	0.907±0.005	0.929±0.006	0.951±0.003
	ONCF	0.875	0.900±0.004	0.918±0.004	0.935±0.007	0.950±0.002
	CML	0.872	0.893±0.004	0.913±0.005	0.930±0.003	0.943±0.005
	HCCF	0.850	0.872±0.006	0.900±0.010	0.932±0.006	0.938±0.012
	NCL	0.843	0.880±0.005	0.902±0.004	0.925±0.004	0.930±0.011
	JNCF	0.866	0.872±0.004	0.903±0.004	0.938±0.005	0.952±0.007
	RNCF	0.873	0.885±0.006	0.904±0.007	0.930±0.003	0.939±0.010
	RRFN	0.860	**0.865±0.003**	**0.872±0.005**	**0.886±0.004**	**0.898±0.012**
Metric	Method	Clean	Sym. 20%	Sym. 40%	Sym. 60%	Sym. 80%
Accuracy ↑	CF	0.201	0.174±0.002	0.160±0.003	0.118±0.006	0.068±0.010
	NCF	0.199	0.174±0.006	0.158±0.005	0.121±0.004	0.053±0.007
	ONCF	0.197	0.172±0.005	0.169±0.003	0.127±0.006	0.066±0.007
	CML	0.191	0.183±0.006	0.157±0.010	0.132±0.006	0.076±0.009
	HCCF	0.205	0.182±0.006	0.170±0.007	0.145±0.018	0.090±0.012
	NCL	0.214	0.172±0.006	0.170±0.004	0.139±0.013	0.085±0.014
	JNCF	0.194	0.176±0.005	0.172±0.006	0.135±0.008	0.083±0.011
	RNCF	0.186	0.170±0.007	0.168±0.007	0.140±0.006	0.081±0.005
	RRFN	0.195	**0.188±0.003**	**0.180±0.002**	**0.165±0.009**	**0.122±0.013**

pairflip noise, our method can achieve great robustness when the noise rate is relatively small. When the noise rate is large, our method again obtains the best results. Besides, for Pair. 30% and 40%, the robustness improvement brought by our method is very clear.

4.4.3 Robusifying Recommendation on Fine-grained Joker Ratings

4.4.3.1 Experimental Setup

Datasets. We exploit original Jester[3] here, which is named Jester-21. We first discretise ratings in Jester for follow-up tasks. We divide the whole Jester data into training data and test data, where the ratio is 99:1. We inject two kinds of rating flip noise into

[3]https://eigentaste.berkeley.edu/dataset/

Table 4.5: Experimental results of evaluations on collaborative filtering recommendations with simulated noisy Jester-21. Pairflip rating flip noise is exploited. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy). The best result in each case is shown in **boldface**.

Metric	Method	Clean	Pair. 20%	Pair. 30%	Pair. 40%	Pair. 45%
MAE ↓	CF	0.865	0.860±0.005	0.868±0.002	0.887±0.005	0.902±0.012
	NCF	0.852	0.862±0.004	0.867±0.006	0.880±0.004	0.890±0.011
	ONCF	0.875	0.891±0.003	0.893±0.005	0.894±0.007	0.897±0.004
	CML	0.872	0.877±0.004	0.881±0.006	0.886±0.004	0.888±0.003
	HCCF	0.850	0.859±0.005	0.863±0.006	0.885±0.015	0.902±0.011
	NCL	0.843	**0.857±0.005**	0.873±0.009	0.892±0.011	0.899±0.011
	JNCF	0.866	0.866±0.005	0.862±0.007	0.879±0.006	0.894±0.009
	RNCF	0.873	0.868±0.007	0.870±0.005	0.892±0.009	0.900±0.010
	RRFN	0.860	**0.857±0.004**	**0.860±0.006**	**0.871±0.008**	**0.875±0.011**
Metric	Method	Clean	Pair. 20%	Pair. 30%	Pair. 40%	Pair. 45%
Accuracy ↑	CF	0.201	0.183±0.007	0.165±0.002	0.152±0.006	0.145±0.013
	NCF	0.199	0.180±0.003	0.167±0.003	0.160±0.005	0.153±0.009
	ONCF	0.197	0.178±0.004	0.176±0.005	0.161±0.004	0.159±0.003
	CML	0.191	0.187±0.004	0.178±0.002	0.167±0.005	0.164±0.004
	HCCF	0.205	0.189±0.003	0.168±0.012	0.158±0.003	0.150±0.014
	NCL	0.214	0.188±0.007	0.172±0.010	0.152±0.015	0.146±0.011
	JNCF	0.194	0.184±0.006	0.165±0.003	0.162±0.006	0.159±0.009
	RNCF	0.186	0.182±0.006	0.160±0.005	0.153±0.012	0.146±0.011
	RRFN	0.195	**0.191±0.003**	**0.185±0.003**	**0.180±0.009**	**0.173±0.008**

the original Jester dataset. Specifically, we employ symmetric noise (abbreviated as Sym.) (Patrini et al., 2017) and pairflip noise (abbreviated as Pair.) (Han et al., 2018). For symmetric noise, the noise rate is set to 20%, 40%, 60%, and 80%, respectively. For pairflip noise, the noise rating is set to 20%, 30%, 40%, and 45%, respectively.

Implementation. We follow the settings of experiments on Jester. We use an Adam optimiser, batch size 256, and a learning rate 0.001. The parameter of dropout in learning user and item presentations is set to 0.5. The dimension of presentations is set to 64 by default. We train recommendation models 10 epochs.

4.4.3.2 Comparisons with Baseline Methods

We report experimental results in Tables 4.4 and 4.5. As can be seen, with the increase in noise levels, the performance of different methods is degraded. Compared with baselines, our method clearly reduces the side effect of rating flip noise, leading to lower MAE and higher accuracy on test data.

4.4.4 Ablation Study

4.4.4.1 Risk-consistent Algorithm vs Classifier-consistent Algorithm

Recall that we employ the importance reweighting technique with the estimated transition matrix to combat rating flip noise, which is *risk-consistent*. Note that the classifier-consistent algorithm Patrini et al. (2017) is also popularly used. Here we include "Forward" Patrini et al. (2017) into comparison. Also, the revision with a slack variable is performed. We conduct experiments on noisy Jester-21 with symmetric noise. As can be seen in Table 4.6, the risk-consistent algorithm works better than the classifier-consistent algorithm whether or not the revision is considered. Also, RRFN which is formed with the risk-consistent algorithm and revision always achieves the best performance.

4.4.4.2 The Influence of Representation Dimension

Note that in representation learning, we learn the representations of both users and items. During this procedure, the representation needs to be determined. Here we employ the dataset MovieLens-100k to demonstrate the influence of the representation dimension. Specifically, we vary the dimensions to 8, 16, 32, and 64, respectively. The noise is set to 30% and 40%. In addition to the proposed method, we also investigate the influence of representation dimension on NCF and ONCF. Other experimental settings

Table 4.6: Experimental results of evaluations on collaborative filtering recommendations with simulated noisy Jester-21. Both the risk-consistent algorithm and classifier-consistent algorithm are considered. Symmetric rating flip noise is exploited. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy). The best result in each case is shown in **boldface**.

Metric	Method	Sym. 20%	Sym. 40%	Sym. 60%	Sym. 80%
MAE ↓	RRFN-Forward w/o revision	0.882±0.004	0.890±0.012	0.920±0.010	0.935±0.008
	RRFN w/o revision	0.871±0.006	0.881±0.015	0.907±0.008	0.920±0.011
	RRFN-Forward	0.872±0.003	0.885±0.006	0.902±0.007	0.913±0.009
	RRFN	**0.865±0.003**	**0.872±0.005**	**0.886±0.004**	**0.898±0.012**
Accuracy ↑	RRFN-Forward w/o revision	0.170±0.006	0.165±0.008	0.142±0.011	0.100±0.017
	RRFN w/o revision	0.182±0.004	0.172±0.004	0.155±0.008	0.112±0.016
	RRFN-Forward	0.176±0.009	0.175±0.012	0.150±0.006	0.107±0.013
	RRFN	**0.188±0.003**	**0.180±0.002**	**0.165±0.009**	**0.122±0.013**

Table 4.7: Experimental results of ablation study about the influence of representation dimension. The noise rate is 30%. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy).

Metric	Method/Dimension	8	16	32	64
MAE ↓	NCF	0.380±0.027	0.372±0.022	0.365±0.041	0.360±0.029
	ONCF	0.362±0.024	0.350±0.018	0.323±0.017	0.320±0.015
	RRFN	0.360±0.015	0.342±0.017	0.330±0.020	0.328±0.016
Accuracy ↑	NCF	0.762±0.019	0.787±0.015	0.797±0.031	0.796±0.017
	ONCF	0.773±0.008	0.782±0.014	0.792±0.030	0.791±0.025
	RRFN	0.790±0.022	0.804±0.017	0.819±0.015	0.820±0.025

are not changed. As can be seen in Tables 4.7 and 4.8, with the increase of dimension (*e.g.,* from 8 to 32), overall, the performance of all methods becomes better. However, when the dimension becomes larger, *i.e.*, 64, the performance does not improve clearly. Sometimes, the performance decreases. The reason is that, although a large dimension may better represent users and items, the existence of rating flip noise makes the precise representation learning hard. Therefore, a large dimension may not always bring better performance.

Table 4.8: Experimental results of ablation study about the influence of representation dimension. The noise rate is 40%. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy).

Metric	Method/Dimension	8	16	32	64
MAE ↓	NCF	0.452±0.019	0.440±0.016	0.424±0.047	0.421±0.026
	ONCF	0.440±0.019	0.438±0.020	0.417±0.026	0.419±0.028
	RRFN	0.430±0.027	0.418±0.016	0.400±0.013	0.409±0.025
Accuracy ↑	RNCF	0.650±0.018	0.661±0.026	0.673±0.044	0.672±0.030
	ONCF	0.650±0.026	0.632±0.018	0.640±0.026	0.630±0.028
	RRFN	0.673±0.012	0.689±0.025	0.707±0.040	0.695±0.017

4.4.4.3 Evaluations with Different Metrics

We before used the metrics MAE and Accuracy in evaluations. To make the comparison more comprehensive, we employ the metrics of recall and precision. Specifically, we conduct experiments on MovieLens-100k. The noise is set to 40%. Experimental results are shown in Table 4.9, which demonstrate the superiority of our method under different metrics.

4.4.4.4 Comparison with SGL

We include SGL Wu et al. (2021b) into baselines for comparison. SGL introduces self-supervised learning to enhance recommendations. We adopt SGL-ED as the instantiation of SGL. Experimental results in Table 4.10 show the effectiveness of the proposed RRFN.

4.4.4.5 The Influence of Different Modules of Our Method

Here we study the influence of different modules of the proposed method and investigate what makes our method successful. Specifically, we conduct experiments on MovieLens-100k with 40% rating flipping noise. The results are shown in Table 4.11, which shows the necessity of several building modules of our method.

Table 4.9: Experimental results of evaluations on collaborative filtering recommendations with simulated noisy MovieLens-100k. The noise is set to 40%. The performance is measured by recall and precision. The best result in each case is shown in **boldface**.

Method/Noise	recall	precision
CF	0.650±0.019	0.670±0.026
NCF	0.662±0.035	0.675±0.054
ONCF	0.655±0.037	0.636±0.029
CML	0.675±0.036	0.702±0.033
HCCF	0.652±0.018	0.676±0.014
NCL	0.650±0.027	0.670±0.029
JNCF	0.642±0.049	0.669±0.050
RNCF	0.693±0.026	0.709±0.017
RRFN	**0.700±0.020**	**0.723±0.025**

Table 4.10: Experimental results of comparing SGL on simulated noisy MovieLens-100k, MovieLens-1M, and Jester-21. The noise is set to 40%. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy). The best result in each case is shown in **boldface**.

Metric	Method/Noise	MovieLens-100k+40%	MovieLens-1M+40%	Jester-21+Sym. 40%
MAE ↓	SGL	**0.390±0.020**	0.404±0.029	0.881±0.007
	RRFN	0.400±0.013	**0.389±0.026**	**0.872±0.005**
Accuracy ↑	SGL	**0.711±0.025**	0.730±0.052	0.171±0.007
	RRFN	0.707±0.040	**0.742±0.038**	**0.180±0.002**

Table 4.11: Experimental results of ablation study about the influence of different modules in our method. The noise rate is 40% on MovieLens-100k. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy).

Metric	Method/Noise	MovieLens-100k+40%
MAE ↓	RRFN	**0.400±0.013**
	RRFN without anchor points	0.407±0.018
	RRFN without anchor points & noise transition matrix	0.417±0.026
Accuracy ↑	RRFN	**0.707±0.040**
	RRFN without anchor points	0.662±0.027
	RRFN without anchor points & noise transition matrix	0.640±0.026

Table 4.12: Experimental results of evaluations on collaborative filtering recommendations with simulated noisy Jester-21. Tridiagonal rating flip noise is exploited. The performance is measured by the mean absolute error (MAE) and rating prediction accuracy (Accuracy). The best result in each case is shown in **boldface**.

Metric	Method	Clean	Trid. 20%	Trid. 30%	Trid. 40%	Trid. 45%
MAE ↓	CF	0.865	**0.853±0.007**	0.875±0.009	0.885±0.004	0.899±0.009
	NCF	0.852	0.860±0.005	0.862±0.008	0.875±0.006	0.893±0.007
	ONCF	0.875	0.889±0.005	0.890±0.005	0.895±0.006	0.900±0.012
	CML	0.872	0.875±0.003	0.881±0.007	0.890±0.014	0.895±0.003
	HCCF	0.850	0.857±0.002	0.861±0.005	0.890±0.012	0.910±0.007
	NCL	0.843	0.857±0.008	0.885±0.010	0.890±0.005	0.902±0.006
	JNCF	0.866	0.872±0.006	0.876±0.004	0.885±0.007	0.899±0.012
	RNCF	0.873	0.875±0.003	0.880±0.004	0.902±0.007	0.909±0.003
	RRFN	0.860	0.855±0.009	**0.859±0.004**	**0.870±0.009**	**0.880±0.012**
Metric	Method	Clean	Trid. 20%	Trid. 30%	Trid. 40%	Trid. 45%
Accuracy ↑	CF	0.201	0.185±0.009	0.172±0.002	0.156±0.009	0.150±0.016
	NCF	0.199	0.182±0.007	0.169±0.006	0.165±0.007	0.150±0.008
	ONCF	0.197	0.182±0.006	0.180±0.006	0.162±0.007	0.152±0.012
	CML	0.191	0.185±0.016	0.180±0.014	0.170±0.003	0.162±0.006
	HCCF	0.205	**0.192±0.007**	0.170±0.014	0.162±0.011	0.152±0.014
	NCL	0.214	0.190±0.001	0.172±0.006	0.155±0.008	0.150±0.012
	JNCF	0.194	0.180±0.007	0.162±0.010	0.162±0.001	0.150±0.013
	RNCF	0.186	0.180±0.003	0.162±0.009	0.158±0.010	0.142±0.007
	RRFN	0.195	**0.192±0.005**	**0.188±0.003**	**0.179±0.008**	**0.168±0.009**

4.4.4.6 Evaluations with other types of noise

We include tridiagonal noise (abbreviated as Trid.) (Zhang et al., 2021b) to verify the effectiveness of the proposed method. We conduct experiments on noisy Jester-21. The noise rating is set to 20%, 30%, 40%, and 45%. We provide experimental results in Table 4.12, which demonstrate the effectiveness of the proposed method.

4.5 Summary

In this Chapter, we focus on enhancing the robustness of recommender systems under rating flip noise. We learn the noise transition matrix to model the flip noise, without knowing the noise rate as a prior. The loss correction methods are applied for the robust

training of recommendation models, which is a statistically risk-consistent algorithm. We demonstrate the state-of-the-art performance of the proposed method over multiple baselines with extensive experiments on multiple noisy datasets.

The current method to model rating flip noise depends only on the rating set. In the future, we will study more complex models to handle the real-world rating flip noise. For example, the noise would depend on the items and users. Moreover, it is also very important to build real-world datasets which contain the rating flip noise.

CHAPTER 5

SEQUENCE UNLEARNING FOR SEQUENTIAL RECOMMENDER SYSTEMS

Sequential recommender systems, leveraging clients' sequential product browsing history, have become an essential tool in delivering personalised product recommendations. As data protection regulations come into focus, certain clients may demand the removal of their data from the training sets used by these systems. In this Chapter, we focus on the problem of how specific client information can be efficiently removed from a pre-trained sequential recommender system without the need for retraining, particularly when the change to the data set is not substantial. We propose a novel sequence unlearning method for sequential recommender systems by leveraging label noise injection. Intuitively, our method promotes data unlearning by encouraging the system to produce random predictions for the sequences aiming to unlearn. To further prevent the model from overfitting an incorrect label, which could lead to substantial changes in its parameters, our method incorporates a dynamic process wherein the incorrect label is continually altered during the learning phase. This effectively encourages the model to lose confidence in the

original label, while also discouraging it from fitting to a specific incorrect label. To the best of our knowledge, this is the first work to tackle the unlearning problem in sequential recommender systems without accessing the remaining data. Our approach is general and can work with any sequential recommender system. Empirically, we demonstrate that our method effectively helps different recommender systems unlearn specific sequential data while maintaining strong generalisation performance on the remaining data across different datasets.

5.1 Introduction

In the rapidly advancing world of machine learning, sequential recommender systems,Äîleveraging clients' sequential product browsing history,Äîhave become indispensable for product recommendations Quadrana et al. (2017); Ma et al. (2019). As data protection regulations gain prominence, an evolving challenge emerges: clients may demand the removal of their data from the training sets used by these systems, often due to concerns over privacy and regulatory compliance Bourtoule et al. (2021).

Existing traditional methods for recommender systems involve retraining the model with the remaining data after excluding the specific client's information Ginart et al. (2019); Bourtoule et al. (2021). However, this retraining process is often resource-intensive and can be unfeasible when the original training data is inaccessible Cha et al. (2023). This leads to the central question explored in this Chapter: How can specific client information be efficiently removed from a pre-trained sequential recommender system without the need for retraining, especially when the change to the dataset is not substantial?

To solve this problem, this Chapter introduces a novel unlearning method through label noise injection based on the learning behaviour of learning models, where learning models usually demonstrate a smaller loss on trained data compared to untrained data.

Specifically, consider two trained models: one trained on a complete dataset and another trained on a pruned dataset excluding some sequences that are aimed to be forgotten. The latter model usually has a higher loss on the to be forgotten sequences compared with the one trained on the complete dataset. This happens because the latter model does not have an opportunity to fit the excluded sequences, making it less confident in predicting them. Motivated by this observation, our proposed method introduces label noise into the to be forgotten sequences. By subsequently fine-tuning a pre-trained model only with these to be forgotten sequences, we amplify the fine-tuned model's loss on the original to be forgotten sequences, making these sequences to be unlearned. This technique effectively emulates the results one might expect from a network retrained on the remaining data without undergoing a full retraining process.

However, the intuition brings in some challenges. Introducing incorrect labels via label-noise injection and fine-tuning a pre-trained model on the noisy examples can lead a model to overfit these incorrect labels, dramatically changing its parameters. This change can dramatically affect the model's performance on the remaining data. This issue is particularly critical in sequential recommender systems, where sequences in training data usually correlate with each other. Then if sequences targeted for removal strongly correlate with some remaining sequences, overfitting an incorrect label could greatly degenerate performance on the remaining data. To mitigate the adverse impacts of label-noise injection, we propose a dynamic process wherein the incorrect label is continually modified during the learning phase. This inventive strategy helps the model predict less confidently on the forgotten sequence, while also protecting it from fitting to a specific incorrect label. To our knowledge, this is the first study to delve into the unlearning issue in sequential recommender systems. Our approach can work with any sequential recommender systems, and presents a comprehensive solution for their unlearning.

5.2 Background

5.2.1 Sequential Recommender Systems

In the early stages of recommender systems development, techniques primarily focused on mapping users and items into latent factor space, with matrix factorisation methods being widely adopted Koren et al. (2009); Lu et al. (2020). As the field evolved, implicit feedback mechanisms, such as purchases or video views, emerged as valuable sources of user information, inspiring new methods designed to harness these insights He et al. (2017); Chen et al. (2020). However, these advancements overlooked one critical aspect: the sequential patterns in consecutively interacted items. Traditional methods, though effective in certain applications, failed to capture this essential aspect of user behaviour. Recognising that sequential patterns in user-item interactions reveal valuable insights into user preferences and evolving interests marked a turning point in the field Shambour and Lu (2015), highlighting the necessity to transition from static models to systems capable of utilising rich sequential information within user interactions.

In response to this need, sequential recommender systems emerged, prioritising recommendations based on a user's historical interaction sequence. Pioneering methods like Markov Chain laid the groundwork for predicting the next item, mainly based on previous interactions, using matrix factorisation Rendle et al. (2010). Deep learning further enriched sequential recommender systems He and McAuley (2016a). For instance, Caser Tang and Wang (2018) applied convolutional neural networks to sequence modelling, treating the item embedding matrix as an image and using convolution operators to identify local transitions. The field's continued evolution integrated Recurrent Neural Networks Quadrana et al. (2017) and Self-Attention mechanisms Kang and McAuley (2018). GRU4Rec Hidasi et al. (2015) leveraged Gated Recurrent Units in session-based recommendations, and the success of methods like Transformer Vaswani et al. (2017) and

BERT Devlin et al. (2018) further promoted self-attention in sequential recommendations. Unlike Markov chain and RNN methods, self-attention considers attention scores from all item-item pairs within a sequence. Both SASRec Kang and McAuley (2018) and BERT4Rec Sun et al. (2019a) highlight the effectiveness of self-attention, achieving leading performance in a next-item recommendation.

5.2.2 Machine Unlearning

Machine unlearning techniques span two primary domains: exact and approximate unlearning. Exact unlearning strategies ensure absolute data forgetting or complete removal Bourtoule et al. (2021), often exploiting a divide-aggregate structure to boost retraining efficiency through model Ginart et al. (2019); Schelter et al. (2021) and data partitioning Yan et al. (2022).

Approximate unlearning, conversely, trades off some level of completeness to enhance efficiency and model utility, thus achieving only statistical forgetting Izzo et al. (2021). These methods often work by directly reducing the target data's influence on the model through inverse gradient-based update strategies, enabling efficient forgetting without retraining. Influence functions, originating from robust statistics Hampel (1974), measure the impact of target data on a trained model, but their application in machine unlearning usually requires the computation of the Hessian matrix, necessitating efficiency optimisation Wu et al. (2022a); Mehta et al. (2022).

5.2.3 Unlearning Method for Recommender Systems

Several unlearning strategies have emerged within the field of recommender systems. RecEraser Chen et al. (2022) implements novel data partition algorithms to divide the training data into balanced groups, preserving collaborative information vital for collaborative filtering. LASER Li et al. (2022a), concurrent with RecEraser, follows a

similar process but with distinct specifics. Various recommendation unlearning works have adopted influence functions from classification or regression tasks Sekhari et al. (2021), using second-order optimisation methods like Newton and quasi-Newton to accelerate re-training Tsai et al. (2014). AltEraser Liu et al. (2022) enhances optimisation efficiency by breaking down a large problem into smaller, independently solvable sub-problems using alternating optimisation. SCIF Li et al. (2023b) selectively updates user embeddings to reduce parameter updates while preserving collaborative information, enhancing model utility. Other notable methods include IFRU Zhang et al. (2023), which extends the influence function, and efforts by Yuan et al. Yuan et al. (2023b) in the federated recommendation. Lastly, Unlearn-ALS Xu et al. (2023b) modifies fine-tuning for bi-linear recommendation models under Alternating Least Squares optimisation. To the best of our knowledge, all existing unlearning methods for recommender systems require remaining data and are not designed specifically for sequential unlearning.

5.3 Sequence Unlearning via Random Label Noise Injection

5.3.1 Problem Setup.

Consider a dataset D of user-item interaction sequences with m users, defined as $D = \{S^{u_1}, S^{u_2}, \ldots, S^{u_m}\}$. Within D, each sequence $S^{u_i} = \{S_1^{u_i}, S_2^{u_i}, \ldots, S_n^{u_i}\}$ is the item interactions of user u_i over n sequential time steps. The primary objective is to predict the coming item in a user's sequence. Specifically, the model, at any time step t, leverages the previous t items from a user's interaction sequence to forecast the next item.

Contrastingly, sequence unlearning seeks to let the model forget a specific training sequence S^{u*}. Specifically, for any time step t, we want the model to generate random predictions for the next item. At the same time, the unlearning should not greatly

impact the model's performance on other interaction sequences, ensuring that its general predictive capabilities remain largely unaffected.

5.3.2 Sequence Unlearning with Dynamic Label Noise Injection

In sequential recommender systems, efficiently unlearning specific sequences without retraining is an emerging critical requirement. To address this challenge, we present a method that dynamically injects label noise into the sequences requested to be removed. Our method centers on noise-generation functions that are crafted to introduce a controlled degree of randomness into the output sequence associated with the sequences the system aims to unlearn. By injecting random label noise into the targeted sequences, the sequential recommender system can be guided to produce less accurate predictions for those sequences.

It is noteworthy that in a sequential recommendation dataset, the sequences in the training data typically exhibit correlations with each other. The label noise may mislead the model's performance on other similar data to the to be forgotten data. To mitigate potential overfitting to these incorrect labels, a situation that could dramatically alter the model's parameters after fine-tuning and negatively impact performance on the remaining data, we propose a dynamic noise injection process. In this process, the incorrect label is continually modified during the learning phase. This innovative approach not only encourages the model to not predict the original label of the to be forgotten data but also ensures that the model does not overfit the label noise. In doing so, we effectively reduce the overfitting problem, maintaining a balance between unlearning specific information and preserving the generalisation ability of the model on remaining data. The pseudocode is summarised in Algorithm 2.

Algorithm 2 Unlearning for Sequential Recommender Systems

Require: Trained recommender systems, User sequence set S, Item set I, To be forgotten set D_f, Noise injection probability ρ

for each epoch **do** {Iterate through training epochs}
  for each S^{u*} in D_f **do** {Iterate through sequences to be forgotten}
    $O^{u*} \leftarrow$ Corresponding output sequence of S^{u*}
    Initialize an empty perturbed label sequence $O^{u*'}$
    for each time step t in O^{u*} **do** {Iterate through each time step }
      Generate a uniform random number x in [0, 1]
      if $x \leq \rho$ **then**
        $s_t^{u*'} \leftarrow$ Random label from $I \setminus \{s_t^{u*}\}$ {To have incorrect labels}
      else
        $s_t^{u*'} \leftarrow s_t^{u*}$ {Retain the original label}
      end if
      $O^{u*'} \leftarrow O^{u*'} \cup [s_t^{u*'}]$ {Update the perturbed label sequence}
    end for
    Update the recommender system using D_f and $O^{u*'}$ {Fine-tune the system with noisy labels}
  end for
end for
=0

5.3.3 Label Noise Injection.

During the training process, at the time step t, the model predicts the next item based on the previous t items. Consider that the model's input for a user-item sequence S^u is $\{s_1^u, s_2^u, \ldots, s_{n-1}^u\}$ and the label sequence (or desired output) is a "shifted" version of the same sequence: $O^u = \{s_2^u, s_3^u, \ldots, s_n^u\}$. For a sequence S^{u*} targeted for unlearning, our method generates a perturbed label sequence $O^{u*'}$ by injecting random label noise into the original label sequence O^{u*}. By fine-tuning the model using the modified label sequences, the model can effectively unlearn these sequences.

Specifically, let I denote the set of all items. We define an incorrect-label-generation function ϕ that, with a probability ρ, replaces an item label with a random one from the set. The perturbed label sequence is generated as:

1. For each time step t in the sequence, generate a random number x uniformly

between 0 and 1.

2. If $x \leq \rho$, apply function ϕ to randomly select an item label from $I \setminus \{s_t^{u*}\}$. If $x > \rho$, keep the original label.

This approach introduces label noise into certain portions of the sequence based on ρ, while the rest remain unchanged. The perturbed label sequence for training are represented as $O^{u*'} = \{s_2^{u*'}, s_3^{u*'}, \ldots, s_n^{u*'}\}$. Each label $s_t^{u*'}$ is defined by:

$$s_t^{u*'} = \begin{cases} \phi(s_t^{u*}), & \text{if } x \leq \rho; \\ s_t^{u*}, & \text{if } x > \rho, \end{cases}$$

where $\phi(s_t^{u*}) \sim U(I \setminus \{s_t^{u*}\})$. Here, $U(I \setminus \{s_t^{u*}\})$ signifies a uniform distribution over item set I excluding s_t^{u*}. By adjusting ρ, we can control the degree of noise infusion and the extent of unlearning, allowing us to find a balance between fulfilling unlearning requests and preserving the system's performance on the remaining data. In our experiment, we set $\rho = 1/p$, where p denotes the total number of items. This enables the model to randomly guess the next item given a sequence aimed to be forgotten.

5.3.4 Unlearning Objective.

The unlearning objective is central to our method, serving to guide the fine-tuning of a pre-trained sequential recommender system for the effective unlearning of particular sequences, while also mitigating potential adverse effects of this unlearning. A distinctive feature of our approach is the dynamic label noise injection. Specifically, we randomly adjust labels continuously throughout each training epoch. This dynamic modification ensures the model does not overfit to any specific incorrect label, preserving the model's generation ability on remaining data.

Let $f_{\hat{\theta}}$ be a pre-trained sequential recommender system. This system outputs probabilities of different items being the next item given a historical user-item interaction

sequence as input. To be precise, given a user's item interaction sequence $\{s_1, s_2, \ldots, s_{t-1}\}$ from the first time step up to $t-1$, the i-th output of the system is $f_{\hat{\theta}}(\{s_1, s_2, \ldots, s_{t-1}\})_i = p_{(s_i,t)}$. Here, $p_{(s_i,t)}$ represents the probability that item s_i will be the next item given the first $t-1$ items in a user-item sequence. We employ the binary cross-entropy loss function to design the objective for fine-tuning:

$$L' = -\sum_{S^{u*} \in D_f} \sum_{t \in \{1,2,\ldots,n\}} \left[\log(p_{(s_t^{u*\prime},t)}) + \sum_{s_i \neq s_t^{u*\prime}} \log(1 - p_{(s_i,t)}) \right],$$

where $s_t^{u*\prime}$ is the randomly picked item via noise injection at time step t given the first $t-1$ items in the sequence S^{u*} targeted for unlearning. The term $p_{(s_t^{u*\prime},t)}$ quantifies the probability that the randomly picked item $s_t^{u*\prime}$ will be the next chosen item as predicted by the pre-trained sequential recommender system. Similarly, $p_{(s_i,t)}$ quantifies the probability that the system predicts an item s_i other than $s_t^{u*\prime}$ as the next item.

Intuitively, minimising the loss function above makes the recommender system more likely to predict randomly picked items by adding label noise and less likely to predict other items. By integrating dynamic label noise with the unlearning objective, our method ensures a small influence on the pre-trained sequential recommender system while achieving the unlearning of specified sequences.

5.4 Experiments

5.4.1 Evaluation Metrics.

Our evaluation framework relies on a selection of widely accepted metrics that comprehensively measure the quality of recommendations. The top-k Hit Ratio (HR@k) represents the average number of positively rated items found in the top-k recommendations for each user. The top-k Normalised Discounted Cumulative Gain (NDCG@k) extends the concept of HR@k by also considering the positions of the positively-rated

items within the top-k list. Unlike HR@k and NDCG@k, which focus on the top-k items, Mean reciprocal rank (MRR) assesses the ranking across the entire list. We have chosen to rank all items, an approach that diverges from biased sampling Krichene and Rendle (2020), and our analysis reports the averaged metrics over all users, where k is set to 10.

5.4.2 Datasets and Experiment Setup.

Our experimental evaluation utilises four renowned benchmark datasets from Amazon reviews, encompassing a wide range of domains. These datasets collectively include more than 1.2 million users and a collection of over 63,000 items. Noted for their significant sparsity, the Amazon datasets furnish a rich compilation of rating reviews spread across various fields. To create sequential patterns, interactions for each user are organised in chronological order based on rating timestamps. Following the methodology in Fan et al. (2022), we designate the most recent interaction as the testing sample, represented by D_t, and the penultimate interaction as the validation sample. We filter to include only users with at least five interactions, aligning with the 5-core setting employed by prominent research in this domain Sun et al. (2019a); Kang and McAuley (2018).

In the unlearning experiment, we randomly select a subset of images from the entire training sample of each dataset to form the unlearning sample D_f. The remaining samples are labelled as D_r. The size of this subset is experimentally determined, with set cardinalities ranging from 5 to 40. The AdaGrad optimiser is engaged to conduct the unlearning process, which is promptly halted if the Mean Reciprocal Rank (MMR) on unlearning sample D_f reaches 0.

5.4.3 Baselines.

In our study, we compare our proposed unlearning methods with several benchmark techniques. Among these baselines, "BEFORE" refers to the model's state before any

unlearning process. The technique "NegGrad" Golatkar et al. (2020) involves fine-tuning the model using negative gradients, while "fixRand" describes a method that introduces random label noise but keeps the labels consistent during the unlearning phase. In contrast, "DyRand," our final approach, injects random label noise and dynamically alters the labels as the unlearning progresses.

We apply these unlearning methods to three distinct sequential recommender systems. Specifically, SASRec Kang and McAuley (2018) is a self-attention-based model that uses a multi-head attention mechanism to recommend the next item. S^3-Rec leverages the intrinsic correlations within the data, enhancing representations through pre-training to improve sequential recommendations. Lastly, STOSA employs a Wasserstein self-attention module, enabling it to characterise item-to-item positional relationships within sequences effectively. These diverse methods and systems offer a rich ground for evaluating the efficacy and adaptability of our proposed unlearning techniques.

Table 5.1: HIT@10, NDCG@10 and MRR by applying different unlearning methods to SASRec on Beauty dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	40.00	30.00	33.33	40.00	40.00	43.33	45.71	45.00
		NDCG@10	18.93	14.46	14.23	15.84	14.98	17.26	18.15	18.10
		MRR	16.66	14.33	12.81	12.37	11.36	13.06	13.31	13.34
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	56.28	56.29	56.30	56.30	56.30	56.30	56.30	56.30
		NDCG@10	27.24	27.24	27.24	27.24	27.25	27.25	27.25	27.25
		MRR	20.79	20.79	20.79	20.79	20.80	20.80	20.80	20.80
	NegGrad	HIT@10	45.03	43.00	32.72	43.67	19.66	41.12	35.66	32.20
		NDCG@10	22.39	21.30	16.25	21.38	10.02	20.35	17.86	16.15
		MRR	17.66	16.85	13.15	16.85	8.37	16.12	14.36	13.04
	FixRand	HIT@10	50.77	51.42	50.91	51.00	49.17	49.65	45.29	47.71
		NDCG@10	24.75	25.51	25.14	25.25	**25.12**	25.28	**25.04**	24.52
		MRR	19.20	19.91	19.61	19.55	19.54	19.74	19.21	19.38
	DyRand	HIT@10	**51.16**	**51.94**	**51.45**	**51.14**	**50.91**	**50.69**	**50.10**	**47.96**
		NDCG@10	**25.23**	**25.72**	**25.45**	**25.42**	25.04	**25.32**	24.99	**24.77**
		MRR	**19.65**	**19.99**	**19.82**	**19.61**	**19.75**	**20.09**	**19.65**	**19.71**
D_t (↑)	BEFORE	HIT@10	5.68	5.68	5.68	5.68	5.68	5.68	5.68	5.68
		NDCG@10	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.03
		MRR	2.50	2.50	2.50	2.50	2.50	2.50	2.50	2.50
	NegGrad	HIT@10	4.94	5.09	4.34	4.90	2.44	4.97	4.22	3.76
		NDCG@10	2.64	2.71	2.26	2.60	1.30	2.66	2.30	2.03
		MRR	2.20	2.23	1.85	2.16	1.11	2.21	1.94	1.68
	FixRand	HIT@10	5.54	5.50	5.37	5.42	5.25	5.23	4.99	**5.04**
		NDCG@10	**2.94**	2.94	2.90	**2.91**	2.86	2.80	2.65	**2.71**
		MRR	**2.40**	2.44	2.43	2.42	2.40	2.33	2.19	**2.27**
	DyRand	HIT@10	**5.59**	**5.52**	**5.42**	**5.42**	**5.35**	**5.29**	**5.25**	4.97
		NDCG@10	**2.94**	**2.97**	**2.91**	2.90	**2.90**	**2.89**	**2.83**	2.69
		MRR	**2.40**	**2.47**	**2.43**	**2.41**	**2.42**	**2.43**	**2.36**	2.25

Table 5.2: HIT@10, NDCG@10 and MRR by applying different unlearning methods to SASRec on Sports and Outdoors dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	0.00	0.00	6.67	5.00	4.00	3.33	2.86	5.00
		NDCG@10	0.00	0.00	4.21	3.15	2.52	2.10	1.80	2.30
		MRR	0.00	0.00	4.42	3.70	2.96	2.72	2.33	2.55
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	6.73	6.73	6.73	6.73	6.73	6.73	6.73	6.73
		NDCG@10	3.33	3.33	3.33	3.33	3.33	3.33	3.33	3.33
		MRR	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10
	NegGrad	HIT@10	6.43	6.21	4.79	4.06	4.26	2.96	3.26	2.74
		NDCG@10	3.28	3.24	2.35	2.36	2.06	1.51	1.67	1.42
		MRR	2.57	2.53	2.13	2.03	1.83	1.33	1.48	1.26
	FixRand	HIT@10	6.68	6.34	6.31	5.82	5.88	5.77	5.26	5.14
		NDCG@10	3.20	3.10	2.96	2.62	2.54	2.53	2.18	2.05
		MRR	3.02	2.93	2.82	2.47	2.39	2.22	2.08	1.92
	DyRand	HIT@10	**6.71**	**6.60**	**6.52**	**6.42**	**6.33**	**5.86**	**5.45**	**5.17**
		NDCG@10	**3.22**	**3.20**	**3.15**	**2.97**	**2.70**	**2.61**	**2.41**	**2.30**
		MRR	**3.10**	**3.03**	**2.98**	**2.86**	**2.61**	**2.43**	**2.22**	**2.18**
D_t (↑)	BEFORE	HIT@10	2.59	2.59	2.59	2.59	2.59	2.59	2.59	2.59
		NDCG@10	1.37	1.37	1.37	1.37	1.37	1.37	1.37	1.37
		MRR	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19
	NegGrad	HIT@10	2.42	2.24	1.72	2.45	1.60	1.46	1.59	1.11
		NDCG@10	1.27	1.19	0.88	1.26	0.80	0.71	0.82	0.55
		MRR	1.11	1.05	0.78	1.08	0.69	0.60	0.71	0.47
	FixRand	HIT@10	**2.57**	2.57	2.38	2.32	2.26	2.16	2.03	1.84
		NDCG@10	1.37	**1.37**	1.26	1.22	1.19	1.14	1.06	0.97
		MRR	1.19	1.19	1.08	1.04	1.00	0.97	0.89	0.82
	DyRand	HIT@10	2.56	**2.58**	**2.39**	**2.54**	**2.28**	**2.31**	**2.16**	**2.01**
		NDCG@10	**1.37**	**1.37**	**1.27**	**1.3**0	**1.19**	**1.17**	**1.11**	**1.02**
		MRR	**1.20**	**1.20**	**1.10**	**1.10**	**1.02**	**0.97**	**0.92**	**0.85**

Table 5.3: HIT@10, NDCG@10 and MRR by applying different unlearning methods to SASRec on Toys and Games dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	20.00	30.00	33.33	35.00	40.00	36.67	34.29	32.50
		NDCG@10	12.62	14.48	18.55	17.84	20.68	18.67	17.81	16.83
		MRR	10.87	11.05	14.99	13.78	16.33	14.92	14.51	13.62
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	36.24	36.24	36.24	36.24	36.23	36.23	36.24	36.24
		NDCG@10	17.46	17.46	17.46	17.46	17.46	17.46	17.46	17.46
		MRR	13.89	13.89	13.88	13.89	13.88	13.88	13.88	13.89
	NegGrad	HIT@10	34.29	29.77	24.46	2.79	5.83	13.37	19.14	3.45
		NDCG@10	16.47	14.48	12.13	1.34	2.76	6.49	9.41	1.62
		MRR	12.80	11.77	9.83	1.17	2.38	5.57	7.81	1.35
	DyRand	HIT@10	**35.49**	**34.39**	34.13	33.29	29.89	24.53	22.83	17.28
		NDCG@10	**16.90**	**16.77**	16.27	16.40	16.34	13.52	12.63	9.57
		MRR	**13.84**	**13.59**	13.95	13.03	13.30	11.11	10.42	7.95
	FixRand	HIT@10	34.89	34.37	**34.22**	**34.02**	**34.13**	**33.39**	**30.03**	**28.58**
		NDCG@10	16.62	16.63	**16.77**	**16.68**	**16.68**	**16.50**	**16.00**	**14.98**
		MRR	13.67	13.38	**13.41**	**13.10**	**13.04**	**13.09**	**13.04**	**12.16**
D_t (↑)	BEFORE	HIT@10	6.71	6.71	6.71	6.71	6.71	6.71	6.71	6.71
		NDCG@10	3.74	3.74	3.74	3.74	3.74	3.74	3.74	3.74
		MRR	3.12	3.12	3.12	3.12	3.12	3.12	3.12	3.12
	NegGrad	HIT@10	6.07	5.91	5.08	1.14	2.28	4.11	4.71	1.46
		NDCG@10	3.33	3.19	2.66	0.58	1.12	2.14	2.44	0.78
		MRR	2.77	2.64	2.20	0.52	0.95	1.80	2.00	0.70
	FixRand	HIT@10	**6.72**	**6.71**	6.14	5.38	5.34	4.84	4.62	3.89
		NDCG@10	3.72	**3.76**	3.44	2.97	2.99	2.66	2.56	2.11
		MRR	3.10	**3.14**	2.89	2.46	2.50	2.20	2.13	1.75
	DyRand	HIT@10	6.70	6.59	**6.33**	**5.83**	**5.85**	**5.77**	**5.26**	**5.22**
		NDCG@10	**3.73**	3.64	**3.46**	**3.20**	**3.26**	**3.14**	**2.95**	**2.90**
		MRR	**3.12**	3.02	**2.86**	**2.66**	**2.73**	**2.58**	**2.49**	**2.43**

Table 5.4: HIT@10, NDCG@10 and MRR by applying different unlearning methods to SASRec on Yelp dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	0.00	10.00	6.67	5.00	4.00	3.33	5.71	5.00
		NDCG@10	0.00	3.56	2.37	1.78	1.42	1.19	3.87	3.39
		MRR	0.00	1.92	1.28	0.96	0.77	0.72	3.56	3.12
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	4.26	4.26	4.26	4.26	4.26	4.26	4.26	4.26
		NDCG@10	2.20	2.20	2.20	2.20	2.20	2.20	2.20	2.20
		MRR	1.96	1.96	1.96	1.96	1.96	1.96	1.96	1.96
	NegGrad	HIT@10	**4.50**	4.02	5.00	4.14	4.82	4.27	3.57	0.38
		NDCG@10	**2.29**	2.07	2.51	2.15	2.49	2.17	1.98	0.21
		MRR	**1.99**	1.86	2.17	1.91	2.21	1.92	1.73	0.20
	FixRand	HIT@10	4.31	5.42	5.03	5.59	5.29	5.86	3.05	3.23
		NDCG@10	2.22	**2.80**	2.57	2.88	2.70	3.06	1.60	1.65
		MRR	1.98	**2.49**	2.32	2.52	2.39	2.71	1.39	1.42
	DyRand	HIT@10	4.31	**5.43**	**5.76**	**5.44**	**5.65**	**6.11**	**4.24**	**4.20**
		NDCG@10	2.22	2.79	**2.98**	**2.81**	**2.93**	**3.20**	**2.18**	**2.15**
		MRR	1.98	**2.49**	**2.63**	**2.50**	**2.59**	**2.81**	**1.87**	**1.85**
D_t (↑)	BEFORE	HIT@10	2.75	2.75	2.75	2.75	2.75	2.75	2.75	2.75
		NDCG@10	1.40	1.40	1.40	1.40	1.40	1.40	1.40	1.40
		MRR	1.20	1.20	1.20	1.20	1.20	1.20	1.20	1.20
	NegGrad	HIT@10	2.26	2.31	2.15	1.98	2.10	2.30	1.25	0.16
		NDCG@10	1.14	1.16	1.09	1.00	1.06	1.14	0.62	0.08
		MRR	0.98	1.01	0.94	0.85	0.91	0.97	0.52	0.08
	FixRand	HIT@10	2.74	2.49	2.50	2.40	2.43	1.91	1.30	1.28
		NDCG@10	**1.39**	1.26	1.27	1.21	1.25	0.82	0.66	**0.64**
		MRR	**1.19**	1.07	1.08	1.03	1.07	0.69	0.57	**0.55**
	DyRand	HIT@10	**2.75**	**2.52**	**2.60**	**2.44**	**2.47**	**2.50**	**1.42**	**1.29**
		NDCG@10	**1.39**	**1.27**	**1.32**	**1.26**	**1.27**	**1.27**	**0.71**	**0.64**
		MRR	**1.19**	**1.09**	**1.13**	**1.09**	**1.10**	**1.08**	**0.61**	**0.55**

Table 5.5: HIT@10, NDCG@10 and MRR by applying different unlearning methods to STOSA on Beauty dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	20.00	30.00	40.00	40.00	40.00	33.33	31.43	32.50
		NDCG@10	12.62	16.18	23.01	22.57	20.80	17.34	16.29	16.34
		MRR	11.59	12.46	18.76	18.03	15.88	13.58	12.59	12.30
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_f (↑)	BEFORE	HIT@10	44.55	44.55	44.55	44.55	44.55	44.56	44.56	44.56
		NDCG@10	27.37	27.37	27.37	27.37	27.37	27.38	27.38	27.39
		MRR	23.13	23.13	23.13	23.13	23.14	23.14	23.14	23.15
	NegGrad	HIT@10	14.10	13.92	14.03	8.88	7.78	7.84	11.13	8.02
		NDCG@10	9.05	8.90	9.05	4.67	3.96	4.00	7.19	4.17
		MRR	7.92	7.78	7.93	3.93	3.28	3.31	6.33	3.49
	FixRand	HIT@10	18.93	19.34	22.16	19.44	20.28	20.49	17.50	16.53
		NDCG@10	12.11	11.88	13.72	12.35	12.71	13.06	11.19	10.50
		MRR	10.54	10.15	11.72	10.69	10.94	11.34	9.73	9.11
	DyRand	HIT@10	**27.74**	**21.52**	**21.18**	**19.63**	**21.50**	**24.57**	**22.02**	**18.43**
		NDCG@10	**17.01**	**13.41**	**13.38**	**12.45**	**13.57**	**15.14**	**13.59**	**11.70**
		MRR	**14.52**	**11.50**	**11.57**	**10.80**	**11.72**	**12.89**	**11.62**	**10.12**
D_t (↑)	BEFORE	HIT@10	6.42	6.42	6.42	6.42	6.42	6.42	6.42	6.42
		NDCG@10	3.71	3.71	3.71	3.71	3.71	3.71	3.71	3.71
		MRR	3.17	3.17	3.17	3.17	3.17	3.17	3.17	3.17
	NegGrad	HIT@10	3.60	3.56	3.55	3.72	3.12	3.23	3.11	3.09
		NDCG@10	2.05	1.98	2.00	1.89	1.56	1.63	1.68	1.60
		MRR	1.78	1.71	1.74	1.57	1.32	1.38	1.45	1.36
	FixRand	HIT@10	4.00	3.35	3.66	3.34	3.51	3.72	2.93	2.63
		NDCG@10	2.27	1.89	2.06	1.90	2.01	2.14	1.69	1.56
		MRR	1.97	1.65	1.78	1.68	1.77	1.89	1.52	1.39
	DyRand	HIT@10	**4.33**	**3.71**	**4.07**	**3.81**	**3.86**	**4.03**	**3.47**	**2.83**
		NDCG@10	**2.54**	**2.11**	**2.31**	**2.13**	**2.23**	**2.32**	**1.97**	**1.61**
		MRR	**2.23**	**1.85**	**2.01**	**1.85**	**1.96**	**2.04**	**1.73**	**1.43**

Table 5.6: HIT@10, NDCG@10 and MRR by applying different unlearning methods to STOSA on Sports and Outdoors dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	20.00	10.00	20.00	15.00	16.00	20.00	17.14	17.50
		NDCG@10	12.62	6.31	15.08	11.31	13.05	15.64	13.41	13.31
		MRR	10.57	6.05	14.43	10.82	13.23	15.19	13.10	13.10
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	24.29	24.29	24.29	24.29	24.29	24.29	24.29	24.30
		NDCG@10	14.36	14.36	14.36	14.36	14.36	14.35	14.36	14.36
		MRR	12.14	12.14	12.14	12.14	12.14	12.14	12.14	12.14
	NegGrad	HIT@10	7.93	8.73	6.03	9.98	8.07	5.40	5.75	4.36
		NDCG@10	4.27	4.76	3.19	6.53	5.24	2.86	3.04	2.26
		MRR	3.64	4.06	2.72	5.78	4.62	2.44	2.60	1.93
	FixRand	HIT@10	**23.86**	**23.97**	10.32	9.88	9.95	9.62	9.51	9.52
		NDCG@10	**14.07**	13.59	6.71	6.40	6.42	6.24	6.16	6.18
		MRR	**15.37**	**13.04**	5.91	5.63	5.64	5.49	5.43	5.44
	DyRand	HIT@10	23.79	23.60	**14.77**	**16.45**	**16.60**	**12.29**	**10.97**	**13.59**
		NDCG@10	13.96	**13.83**	**10.44**	**10.43**	**9.41**	**7.91**	**7.04**	**8.53**
		MRR	15.25	12.81	**9.10**	**9.03**	**8.24**	**6.94**	**6.20**	**7.42**
D_t (↑)	BEFORE	HIT@10	3.43	3.43	3.43	3.43	3.43	3.43	3.43	3.43
		NDCG@10	1.92	1.92	1.92	1.92	1.92	1.92	1.92	1.92
		MRR	1.64	1.64	1.64	1.64	1.64	1.64	1.64	1.64
	NegGrad	HIT@10	2.94	2.21	1.65	1.42	1.46	1.33	1.05	0.97
		NDCG@10	1.58	1.73	1.33	1.21	1.02	0.95	0.64	0.58
		MRR	1.36	1.18	1.13	1.09	0.91	0.82	0.57	0.51
	FixRand	HIT@10	3.30	**2.29**	1.21	1.14	1.16	1.13	1.11	1.12
		NDCG@10	1.93	**1.41**	0.73	0.69	0.71	0.68	0.68	0.68
		MRR	1.67	**1.24**	0.65	0.62	0.63	0.60	0.61	0.60
	DyRand	HIT@10	**3.31**	2.27	**1.87**	**1.56**	**1.64**	**1.60**	**1.42**	**1.55**
		NDCG@10	**1.94**	1.37	**1.33**	**1.27**	**1.21**	**1.00**	**0.85**	**0.93**
		MRR	**1.68**	1.20	**1.14**	**1.11**	**1.03**	**0.90**	**0.75**	**0.82**

Table 5.7: HIT@10, NDCG@10 and MRR by applying different unlearning methods to STOSA on Toys and Games dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	20.00	30.00	26.67	35.00	40.00	36.67	34.29	32.50
		NDCG@10	12.62	15.18	16.79	18.91	24.67	22.67	22.28	20.58
		MRR	10.51	10.87	14.39	15.26	21.28	19.67	20.06	18.46
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	46.20	46.21	46.21	46.21	46.21	46.21	46.22	46.23
		NDCG@10	29.76	29.76	29.76	29.76	29.76	29.76	29.77	29.77
		MRR	25.61	25.61	25.61	25.61	25.61	25.61	25.61	25.62
	NegGrad	HIT@10	7.92	9.12	14.12	8.95	18.26	14.68	15.52	14.38
		NDCG@10	4.47	5.08	8.12	5.09	11.23	9.49	10.11	9.45
		MRR	3.89	4.35	6.95	4.40	9.74	8.27	8.86	8.29
	FixRand	HIT@10	28.34	22.44	22.20	20.40	17.33	17.01	17.40	16.98
		NDCG@10	18.22	14.76	14.60	13.48	11.00	10.89	11.19	10.97
		MRR	15.78	12.87	12.72	11.80	9.50	9.44	9.73	9.57
	DyRand	HIT@10	**32.00**	**29.45**	**25.37**	**24.73**	**20.67**	**21.79**	**20.26**	**18.71**
		NDCG@10	**20.39**	**18.63**	**16.66**	**16.06**	**13.26**	**14.18**	**13.08**	**11.96**
		MRR	**17.57**	**15.97**	**14.52**	**13.96**	**11.50**	**12.35**	**11.38**	**10.38**
D_t (↑)	BEFORE	HIT@10	6.89	6.89	6.89	6.89	6.89	6.89	6.89	6.89
		NDCG@10	4.20	4.20	4.20	4.20	4.20	4.20	4.20	4.20
		MRR	3.62	3.62	3.62	3.62	3.62	3.62	3.62	3.62
	NegGrad	HIT@10	3.49	3.86	5.20	4.08	**5.22**	3.90	4.06	3.64
		NDCG@10	1.92	2.18	3.03	2.25	3.00	2.18	2.36	2.06
		MRR	1.62	1.87	2.60	1.90	2.55	1.83	2.02	1.76
	FixRand	HIT@10	5.58	4.77	4.65	4.67	4.37	4.32	4.36	4.38
		NDCG@10	3.46	2.94	2.90	2.86	2.60	2.57	2.63	2.59
		MRR	3.01	2.56	2.53	2.48	2.25	2.22	2.28	2.21
	DyRand	HIT@10	**5.79**	**5.11**	**4.89**	**4.94**	**4.92**	**4.94**	**4.66**	**4.70**
		NDCG@10	**3.58**	**3.21**	**3.05**	**3.12**	**3.04**	**3.06**	**2.81**	**2.83**
		MRR	**3.12**	**2.81**	**2.66**	**2.73**	**2.64**	**2.67**	**2.43**	**2.43**

Table 5.8: HIT@10, NDCG@10 and MRR by applying different unlearning methods to STOSA on Yelp dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	0.00	10.00	6.67	5.00	4.00	10.00	11.43	10.00
		NDCG@10	0.00	3.01	2.01	1.51	1.20	5.34	7.43	6.51
		MRR	0.69	1.46	0.97	1.01	0.80	4.37	6.84	5.99
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	8.88	8.88	8.88	8.88	8.88	8.87	8.87	8.87
		NDCG@10	5.04	5.04	5.04	5.04	5.04	5.04	5.03	5.04
		MRR	4.39	4.39	4.39	4.39	4.39	4.39	4.38	4.38
	NegGrad	HIT@10	6.53	6.05	6.02	5.53	6.48	4.51	5.93	5.72
		NDCG@10	4.70	4.96	4.53	3.32	4.01	2.32	3.67	3.47
		MRR	4.94	4.33	4.83	2.89	3.50	2.02	3.23	3.02
	FixRand	HIT@10	8.23	8.63	8.81	8.37	8.13	**8.60**	8.04	8.22
		NDCG@10	5.06	**5.55**	5.51	**5.92**	4.99	**5.26**	4.94	5.00
		MRR	4.26	**4.56**	4.62	4.54	4.38	**4.59**	4.33	4.35
	DyRand	HIT@10	**8.54**	**8.71**	**8.92**	**8.63**	**8.86**	8.52	**8.43**	**8.67**
		NDCG@10	**5.85**	5.12	**5.70**	**5.64**	**5.39**	5.19	**5.13**	**5.25**
		MRR	**4.78**	4.02	**4.67**	**4.81**	**4.70**	4.52	**4.47**	**4.56**
D_t (↑)	BEFORE	HIT@10	2.96	2.96	2.96	2.96	2.96	2.96	2.96	2.96
		NDCG@10	1.49	1.49	1.49	1.49	1.49	1.49	1.49	1.49
		MRR	1.27	1.27	1.27	1.27	1.27	1.27	1.27	1.27
	NegGrad	HIT@10	0.80	0.64	0.69	0.36	0.44	2.94	0.38	0.38
		NDCG@10	0.40	0.32	0.34	0.17	0.21	1.48	0.19	0.18
		MRR	0.35	0.29	0.31	0.16	0.19	1.27	0.18	0.16
	FixRand	HIT@10	**1.91**	1.45	1.49	1.59	0.66	0.66	0.62	0.64
		NDCG@10	**1.03**	0.73	0.75	0.78	0.33	0.33	0.29	0.31
		MRR	**0.92**	0.64	0.65	0.68	0.29	0.30	0.26	0.28
	DyRand	HIT@10	1.66	**1.86**	**1.53**	1.47	**0.78**	**0.70**	**0.66**	**0.76**
		NDCG@10	0.81	**0.91**	**0.77**	0.72	**0.40**	**0.35**	**0.33**	**0.37**
		MRR	0.70	**0.78**	**0.67**	0.62	**0.36**	**0.31**	**0.29**	**0.33**

Table 5.9: HIT@10, NDCG@10 and MRR by applying different unlearning methods to S^3Rec on Beauty dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	20.00	30.00	40.00	40.00	40.00	36.67	34.29	32.50
		NDCG@10	20.00	24.31	29.66	28.55	25.90	23.25	21.16	21.01
		MRR	20.80	23.35	27.80	26.11	22.68	20.01	17.87	18.30
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_f (↑)	BEFORE	HIT@10	40.67	40.67	40.66	40.66	40.66	40.67	40.67	40.68
		NDCG@10	25.41	25.41	25.41	25.41	25.41	25.41	25.42	25.42
		MRR	21.69	21.69	21.69	21.69	21.69	21.69	21.70	21.70
	NegGrad	HIT@10	15.94	16.32	14.17	12.67	14.89	13.35	11.67	12.53
		NDCG@10	10.50	10.63	9.25	8.19	9.76	8.84	7.75	8.31
		MRR	9.23	9.28	8.11	7.16	8.57	7.79	6.84	7.32
	FixRand	HIT@10	18.57	20.09	18.17	17.98	18.61	17.31	17.32	17.06
		NDCG@10	12.12	12.92	11.87	11.80	12.23	11.41	11.38	11.23
		MRR	10.58	11.19	10.35	10.33	10.69	10.01	9.96	9.84
	DyRand	HIT@10	**24.08**	**22.93**	**21.55**	**25.09**	**28.84**	**24.08**	**19.01**	**18.68**
		NDCG@10	**15.30**	**14.63**	**13.86**	**16.21**	**18.05**	**15.45**	**12.30**	**12.12**
		MRR	**13.17**	**12.59**	**11.96**	**13.98**	**15.36**	**13.30**	**10.67**	**10.52**
D_t (↑)	BEFORE	HIT@10	6.39	6.39	6.39	6.39	6.39	6.39	6.39	6.39
		NDCG@10	3.63	3.63	3.63	3.63	3.63	3.63	3.63	3.63
		MRR	3.09	3.09	3.09	3.09	3.09	3.09	3.09	3.09
	NegGrad	HIT@10	3.21	3.20	2.88	2.59	3.04	2.84	2.46	2.71
		NDCG@10	1.81	1.82	1.63	1.50	1.75	1.61	1.42	1.52
		MRR	1.57	1.59	1.42	1.33	1.54	1.42	1.25	1.32
	DyRand	HIT@10	**3.67**	2.88	3.11	3.28	3.34	3.10	2.97	2.98
		NDCG@10	**2.11**	1.67	1.79	1.89	1.94	1.80	1.70	1.73
		MRR	**1.84**	1.44	1.57	1.64	1.71	1.59	1.48	1.53
	FixRand	HIT@10	3.46	**3.24**	**3.19**	**3.39**	**3.69**	**3.70**	**3.09**	**3.12**
		NDCG@10	2.04	**1.88**	**1.85**	**2.00**	**2.18**	**2.18**	**1.77**	**1.80**
		MRR	1.79	**1.63**	**1.61**	**1.75**	**1.91**	**1.92**	**1.52**	**1.56**

Table 5.10: HIT@10, NDCG@10 and MRR by applying different unlearning methods to S^3Rec on Sports and Outdoors dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	0.00	0.00	20.00	15.00	12.00	13.33	11.43	15.00
		NDCG@10	0.00	0.00	15.56	11.67	9.33	11.11	9.52	13.33
		MRR	1.19	0.60	14.68	11.52	9.53	11.50	10.05	13.88
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	20.57	20.57	20.56	20.57	20.57	20.57	20.57	20.57
		NDCG@10	12.45	12.45	12.45	12.45	12.45	12.45	12.45	12.45
		MRR	10.70	10.70	10.69	10.70	10.70	10.70	10.70	10.69
	NegGrad	HIT@10	12.39	6.33	5.05	11.26	6.13	7.89	5.98	8.23
		NDCG@10	6.91	3.35	2.71	6.22	3.25	5.23	3.21	5.38
		MRR	5.88	2.87	2.34	5.31	2.80	4.63	2.77	4.75
	DyRand	HIT@10	17.35	17.06	11.85	10.52	10.92	10.69	10.66	10.37
		NDCG@10	**11.59**	11.39	7.85	6.97	7.23	7.09	7.06	6.90
		MRR	8.43	8.14	6.94	6.17	6.40	6.29	6.25	6.12
	FixRand	HIT@10	**17.81**	**17.73**	**16.70**	**13.20**	**13.27**	**12.24**	**11.57**	**11.92**
		NDCG@10	11.50	**11.25**	**10.59**	**8.72**	**8.67**	**8.06**	**7.66**	**7.82**
		MRR	**8.98**	**8.25**	**8.21**	**7.72**	**7.63**	**7.12**	**6.77**	**6.89**
D_t (↑)	BEFORE	HIT@10	3.16	3.16	3.16	3.16	3.16	3.16	3.16	3.16
		NDCG@10	1.75	1.75	1.75	1.75	1.75	1.75	1.75	1.75
		MRR	1.51	1.51	1.51	1.51	1.51	1.51	1.51	1.51
	NegGrad	HIT@10	2.14	1.83	1.61	1.29	1.33	1.24	1.20	1.20
		NDCG@10	1.23	1.11	0.94	0.78	0.80	0.73	0.71	0.72
		MRR	1.05	0.98	0.82	0.69	0.70	0.64	0.63	0.64
	FixRand	HIT@10	**3.11**	2.53	1.24	1.09	1.16	1.13	1.12	1.11
		NDCG@10	**1.77**	**1.54**	0.96	0.66	0.69	0.68	0.67	0.66
		MRR	**1.52**	**1.28**	0.75	0.60	0.61	0.60	0.60	0.59
	DyRand	HIT@10	3.02	**2.76**	**2.35**	**2.40**	**2.58**	**2.57**	**2.54**	**1.97**
		NDCG@10	1.65	1.46	**1.24**	**1.31**	**1.36**	**1.36**	**1.34**	**1.30**
		MRR	1.42	1.26	**1.08**	**1.15**	**1.17**	**1.16**	**1.16**	**0.90**

Table 5.11: HIT@10, NDCG@10 and MRR by applying different unlearning methods to S^3Rec on Toys and Games dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	20.00	40.00	33.33	40.00	44.00	46.67	45.71	47.50
		NDCG@10	12.62	22.65	21.77	21.91	27.25	28.29	28.91	28.88
		MRR	13.50	19.75	20.34	18.80	24.29	24.57	25.68	24.83
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	48.43	48.42	48.43	48.43	48.42	48.42	48.42	48.42
		NDCG@10	30.72	30.72	30.72	30.73	30.72	30.72	30.72	30.72
		MRR	26.23	26.23	26.23	26.24	26.23	26.23	26.23	26.23
	NegGrad	HIT@10	9.63	7.41	18.53	7.29	17.43	13.38	19.24	16.09
		NDCG@10	5.31	4.07	12.15	3.96	10.13	7.55	11.59	9.21
		MRR	4.60	3.56	10.59	3.43	8.60	6.46	9.83	7.80
	FixRand	HIT@10	27.19	25.62	23.47	21.39	21.60	21.46	21.49	21.14
		NDCG@10	17.93	16.90	15.68	14.43	14.42	14.32	14.29	14.11
		MRR	15.64	14.71	13.72	12.72	12.63	12.55	12.48	12.36
	DyRand	HIT@10	**30.10**	**35.21**	**27.55**	**30.63**	**24.66**	**24.19**	**26.08**	**23.85**
		NDCG@10	**19.60**	**22.64**	**18.02**	**19.74**	**16.17**	**15.84**	**16.98**	**15.32**
		MRR	**17.00**	**19.40**	**15.63**	**16.99**	**14.06**	**13.77**	**14.73**	**13.21**
D_t (↑)	BEFORE	HIT@10	6.74	6.74	6.74	6.74	6.74	6.74	6.74	6.74
		NDCG@10	4.08	4.08	4.08	4.08	4.08	4.08	4.08	4.08
		MRR	3.52	3.52	3.52	3.52	3.52	3.52	3.52	3.52
	NegGrad	HIT@10	3.83	3.12	4.15	3.16	3.84	3.70	3.90	3.61
		NDCG@10	2.09	1.72	2.46	1.71	2.22	2.18	2.29	2.12
		MRR	1.76	1.46	2.08	1.44	1.87	1.88	1.95	1.82
	FixRand	HIT@10	5.35	5.00	4.73	4.44	4.60	4.58	4.55	4.49
		NDCG@10	3.33	3.14	2.98	2.79	2.81	2.82	2.81	2.78
		MRR	2.90	2.72	2.59	2.44	2.42	2.43	2.44	**2.42**
	DyRand	HIT@10	**5.70**	**5.66**	**5.11**	**5.24**	**4.82**	**4.73**	**4.98**	**4.57**
		NDCG@10	**3.49**	**3.53**	**3.24**	**3.29**	**3.03**	**2.98**	**3.13**	**2.81**
		MRR	**3.01**	**3.05**	**2.83**	**2.86**	**2.66**	**2.61**	**2.74**	**2.42**

Table 5.12: HIT@10, NDCG@10 and MRR by applying different unlearning methods to S^3Rec on Yelp dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	0.00	10.00	6.67	5.00	4.00	6.67	8.57	7.50
		NDCG@10	0.00	3.33	2.22	1.67	1.33	2.07	4.64	4.06
		MRR	0.87	1.86	1.69	1.56	1.25	1.49	4.14	3.62
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	11.53	11.52	11.53	11.53	11.53	11.53	11.53	11.53
		NDCG@10	6.61	6.61	6.61	6.61	6.62	6.62	6.61	6.61
		MRR	5.71	5.71	5.71	5.71	5.71	5.71	5.71	5.71
	NegGrad	HIT@10	6.10	5.62	5.95	3.85	5.23	4.92	8.77	6.18
		NDCG@10	6.37	5.40	3.53	2.05	3.28	2.81	5.47	3.83
		MRR	3.57	3.73	3.03	1.76	2.86	2.39	4.79	3.33
	FixRand	HIT@10	**8.73**	8.24	7.99	8.29	8.26	**8.31**	8.21	8.05
		NDCG@10	**5.32**	5.03	5.07	5.26	5.27	**5.30**	5.24	5.12
		MRR	**5.05**	4.31	4.45	4.60	4.62	**4.65**	4.60	4.48
	DyRand	HIT@10	8.66	**8.54**	**8.24**	**8.56**	**8.43**	8.18	**8.69**	**8.62**
		NDCG@10	5.19	**5.39**	**5.21**	**5.44**	**5.35**	5.21	**5.51**	**5.45**
		MRR	4.96	**4.86**	**4.56**	**4.75**	**4.69**	4.56	**4.82**	**4.76**
D_t (↑)	BEFORE	HIT@10	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15
		NDCG@10	1.61	1.61	1.61	1.61	1.61	1.61	1.61	1.61
		MRR	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39
	NegGrad	HIT@10	0.45	0.60	0.36	0.35	0.36	0.36	0.37	0.34
		NDCG@10	0.22	0.30	0.17	0.16	0.17	0.16	0.18	0.16
		MRR	0.21	0.27	0.16	0.15	0.15	0.15	0.17	0.15
	FixRand	HIT@10	**2.47**	1.04	0.37	0.35	0.35	0.36	0.35	0.33
		NDCG@10	**1.18**	0.52	0.16	0.15	0.16	0.16	0.15	0.14
		MRR	**0.99**	0.48	0.15	0.14	0.15	0.14	0.14	0.14
	DyRand	HIT@10	2.00	**1.30**	**0.54**	**0.87**	**0.59**	**0.60**	**0.67**	**0.36**
		NDCG@10	0.98	**0.65**	**0.26**	**0.44**	**0.23**	**0.30**	**0.32**	**0.17**
		MRR	0.84	**0.59**	**0.23**	**0.37**	**0.22**	**0.27**	**0.29**	**0.16**

5.4.4 Unlearning Performance on Different Datasets

We evaluate the unlearning performance of various methods applied to different datasets and using different sequential recommender systems. We present results for four datasets (Toys and Games, Yelp, Beauty, Sport and Outdoors) employing the SASRec sequential recommender system within the main text.

Tables 5.1 - 5.12 all exhibit similar findings. All baseline methods successfully unlearn the targeted sample D_f, reflected in a score of 0 across different evaluation metrics. However, a clear distinction emerges when we consider the remaining sample D_r and the

test sample D_t. Both FixRand and DyRand, methods that employ label noise injection to unlearn samples, consistently surpass the NegGrad baseline across all tests.

Notably, the performance of the baseline NegGrad on both D_r and D_t diminishes sharply with increasing unlearning sample size. On the other hand, our newly proposed unlearning methods that use label noise injection remain resilient in performance. Our final approach, dynamic label noise injection (DyRand), stands out by achieving the best performance across various metrics and unlearning sample sizes in the majority of experiments. These results firmly validate our method's capacity to unlearn specific sequential data while retaining the model's predictive competence over the remaining information.

5.5 Summary

In this Chapter, we tackled the challenge of sequence unlearning in sequential recommender systems, a vital area in the age of data protection and privacy. We introduced a new unlearning method leveraging label noise injection. Our approach ingeniously encourages the system to make random predictions for sequences to be unlearned, using a dynamic noise-injection approach to prevent overfitting to incorrect labels and dramatic changing of the model parameter. Empirical results demonstrated effectiveness in unlearning specific data while preserving generalization performance.

CHAPTER 6

TOWARDS SAFE MACHINE UNLEARNING: A PARADIGM THAT MITIGATES PERFORMANCE DEGRADATION

We study the machine unlearning problem which aims to remove specific training data from a pre-trained machine learning model to allow users to exercise their 'right to be forgotten' to protect user privacy. Conventional machine unlearning methods would degrade the model performance after the unlearning procedure. To mitigate the issue, they typically rely on the access to the remaining training data to fine-tune the unlearned model to mitigate the influence of unlearning. However, accessing the remaining training data may not always be practical for different reasons (e.g., data expiration policies, storage limitations, or additional privacy constraints). Machine unlearning without access to the remaining training data poses significant challenges to retaining model performance. In this Chapter, we study how to unlearn specific training data from a pre-trained model without accessing the remaining training data and protect model performance without dramatically changing the model's parameters. We propose a practical method called Targeted Label Noise Injection. Intuitively, our method assigns

incorrect yet controllable labels to the examples that need to be forgotten and fine-tunes the pre-trained model to learn these new labels. This strategy effectively moves the to-be-forgotten examples across the decision boundary with a small impact on the model's overall performance. We theoretically prove the effectiveness of the proposed method and empirically show that it achieves state-of-the-art unlearning performance across various datasets. We also deployed the proposed methods for sequential unlearning for sequential recommender systems.

6.1 Introduction

In the era of big data, machine learning models have become integral to a wide range of web applications, from personalised recommendations to critical decision-making systems. These models are often trained on vast amounts of user data, much of which may be sensitive or personal. As concerns about data privacy escalate, stringent regulations such as the General Data Protection Regulation (GDPR) (Regulation, 2018) have been enacted, granting individuals the right to have their personal data erased/forgotten. There is a growing need for machine learning models to not only learn from data but also to unlearn specific data points upon request.

Machine unlearning (Cao and Yang, 2015) aiming to remove specific data from trained machine learning models has thus emerged as a crucial capability. This ensures compliance with data privacy laws and maintains user trust in systems. Unfortunately, the unlearning procedure would degrade model performance. To address this issue, conventional methods typically require access to both the data to be forgotten and the remaining training data, using the remaining data to tune the unlearned model to mitigate performance degradation (Brophy and Lowd, 2021; Chundawat et al., 2022; Ye et al., 2022). However, accessing the remaining data can be impractical due to many reasons, e.g., data expiration policies that mandate deletion after a certain period, storage

limitations that prevent retaining large datasets, or additional privacy constraints that prohibit the reuse of data without explicit consent.

Unlearning without access to the remaining data poses significant challenges. Existing machine unlearning methods in this setting (e.g., (Li et al., 2021; Ebrahimi et al., 2020; Chen et al., 2023)) are limited and often lead to substantial performance degradation. Intuitively, fine-tuning solely on the examples to be forgotten may require extensive parameter adjustments, resulting in overfitting the to-be-forgotten examples and underperformance on the remaining examples. Motivated by these challenges, in this Chapter, we study how to safely unlearn some examples from a pre-trained model without accessing the remaining training examples and without dramatically changing the model's parameters.

To address this problem, we propose a novel method called Targeted Label Noise Injection for machine unlearning. The key philosophy of our method is to minimise parameter changes by selecting incorrect yet easy-to-learn labels for the to-be-forgotten examples. Specifically, we assign each instance to be forgotten an incorrect label that the model is already somewhat inclined toward. This ensures that the incorrect labels align relatively well with the model's existing knowledge and is easy for the model to learn. By doing so, we reduce the need for large parameter updates, helping to maintain the model's performance on the remaining examples.

However, it is also important to note that during fine-tuning, the model can overfit the pseudo-labels with unnecessarily high confidence. This leads to unwanted parameter adjustments. To prevent overfitting the pseudo-labels, we reduce the contribution of each to-be-forgotten example to the loss once the model starts predicting its pseudo-label correctly. This is achieved by introducing an exponent λ to the training loss. By setting a high value for λ during the fine-tuning process, the loss of instances with pseudo-labels, which the model has already learned, is dramatically reduced. Consequently, in the

later fine-tuning epochs, these instances have little influence on the model's parameter updates, effectively avoiding unnecessary changes to the model's parameters.

Furthermore, we theoretically analyse the parameter changes induced by our method and find that it concentrates updates on a specific subset of parameters associated with to-be-forgotten examples. This targeted adjustment effectively modifies only the necessary parameters while minimising the impact on unrelated parts of the model. Empirically, we have validated that our method achieves the smallest parameter changes across different datasets when compared with various baselines. Our method consistently achieves state-of-the-art performance for machine unlearning without accessing the remaining data, demonstrating its effectiveness and practicality across diverse scenarios.

6.2 Background

We study the machine unlearning aiming to enable a pre-trained model to discard information acquired from a subset of data. The primary objective has been to undo the influence of undesired data on the model while maintaining the model's predictive power on the remaining sample. Historically, simpler machine learning models, such as linear/logistic regression, k-means clustering, and random forests, have been the subjects of various unlearning techniques (e.g., (Mahadevan and Mathioudakis, 2021; Ginart et al., 2019; Mehta et al., 2022)). However, these methodologies are inherently designed for these less complex models and don't seamlessly translate to intricate architectures like deep neural networks.

To achieve unlearning for deep neural networks, many methods have also been proposed. Some of them focus on making the model forget all examples corresponding to a specific class, like all images of a particular category in a dataset (Chundawat et al., 2022; Ye et al., 2022; Yoon et al., 2022). Some of them emphasise erasing information from some examples potentially spanning multiple classes from the pre-trained model (Shibata

et al., 2021; Poppi et al., 2024;). By considering that data deletion may requests are practically received on a per-instance basis, some methods which allow to forget any subset of training examples have also been proposed (Han et al., 2024; Foster et al., 2024; Yu et al., 2022). A notable limitation shared by most method is their reliance on access to both the data intended for unlearning and insights on the remaining training data during the unlearning phase (e.g., (Lee and Woo, 2023; Kim and Woo, 2022)). This dependency becomes problematic in real-world scenarios where accessing the remaining sample might be hindered by factors like data expiration policies or other regulations.

To circumvent this challenge, the use of gradient ascent in the unlearning process has been naturally proposed. The basic idea is to employ gradient ascent to reverse the loss value of an example marked for unlearning from a pre-trained model. However, while the gradient ascent-based unlearning method has demonstrated its effectiveness across diverse settings and datasets (Wu et al., 2022b; Setlur et al., 2022; Jia et al., 2023), concerns have been raised about its suitability in scenarios demanding high privacy and utility (Ye and Lu, 2023; Lu et al., 2015; Lu, 2004). Graves et al. (Graves et al., 2021) argued that merely amplifying loss using gradient ascent doesn't inherently ensure heightened privacy. Suriyakumar and Wilson (Suriyakumar and Wilson, 2022) further highlighted the potential for data leakage even when the loss is accentuated. This brings to light the privacy implications of deploying existing techniques. Some methods have also been proposed which use gradient descent (e.g., (Jia et al., 2023; Tarun et al., 2023; Li et al., 2023a)). However, fine-tuning solely on the examples to be forgotten may require extensive parameter adjustments. This leads to overfitting to the to-be-forgotten examples and underperformance on remaining examples. Motivated by these challenges, we study how to safely unlearn examples from a pre-trained model without accessing the remaining training examples and without dramatically changing the model's parameters. Note that the idea of injecting noise has been used to protect privacy (Abadi et al., 2016),

enhance unlearning (Cha et al., 2024), and increase robustness (Di et al., 2024). We are the first to inject flipping label noise (Natarajan et al., 2013; Ye and Lu, 2024; Wei et al., 2025) to mitigate performance degradation for machine unlearning.

6.3 δ-Targeted$^{\lambda}$ Label Noise Injection for Machine Unlearning

Problem Setup. Consider a dataset $\boldsymbol{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i \in X$ is an instance from the feature space, $y_i \in \{1, 2, \ldots, C\}$ is its corresponding label from C classes, n is the size of the dataset. Each example $(\boldsymbol{x}_i, y_i)$ is assumed to be independently drawn from an underlying distribution $P(\boldsymbol{X}, Y)$.

Let $\sigma \circ f_{\hat{\theta}} : X \to \mathbb{R}^C$ be a pre-trained model trained on $\boldsymbol{D}$ by minimizing the average loss, where $f_{\hat{\theta}}$ represents the deep network and $\hat{\theta}$ denotes the trained parameters, and σ is the softmax function. Given an instance $\boldsymbol{x}$, the model outputs $\sigma \circ f_{\hat{\theta}}(\boldsymbol{x})$ estimates the class-posterior distribution.

Suppose we have a subset $\boldsymbol{D}_f \subseteq \boldsymbol{D}$ of examples that we aim to forget from the model. The remaining examples are denoted as $\boldsymbol{D}_r = \boldsymbol{D} \setminus \boldsymbol{D}_f$. Our objective is to obtain a model that effectively forgets $\boldsymbol{D}_f$ while maintaining high accuracy on $\boldsymbol{D}_r$ and generalisation, using only $\boldsymbol{D}_f$ and the pre-trained model $\sigma \circ f_{\hat{\theta}}$.

6.3.1 Targeted Label Noise Injection with Probability δ

Our method is based on injecting label noise into the to-be-forgotten examples. Intuitively, to mitigate performance degradation, we select incorrect yet easy-to-learn labels for the to-be-forgotten examples. This approach ensures that the incorrect label aligns relatively well with the model's existing knowledge and is easy to learn. This effectively reduces the need for large parameter changes and helps to maintain the model's performance.

Specifically, for each to-be-forgotten example $(\boldsymbol{x}, y) \in \boldsymbol{D}_f$, we compute the model's predicted probabilities $\sigma \circ f_{\hat{\theta}}(\boldsymbol{x})$. Let $\hat{y} = \arg\max_{i \in \{1,2,\ldots,C\}} \sigma \circ f_{\hat{\theta}}(\boldsymbol{x})$ be the predicted label. The second most confident label is defined by $\hat{y}' = \arg\max_{i \in \{1,2,\ldots,C\} \setminus \hat{y}} \sigma \circ f_{\hat{\theta}}(\boldsymbol{x})$. It is easy to find that the second most confident label of a to-be-forgotten example is an easy-to-learn label for it. In this Chapter, we use the second most confident labels as the easy-to-learn labels.

By considering that different scenarios may require different degrees of forgetting, we might want to control the trade-off between the level of unlearning and the preservation of the model's performance. Additionally, if all to-be-forgotten examples are always assigned incorrect labels, an adversary could infer that the true labels are different from the predicted ones, thus gaining information about the original data. Therefore, we introduce a parameter $\delta \in [0,1]$ to control the strength of the label perturbation. Specifically, if the model's predicted label $\hat{y}$ does not match the original label y, this indicates a misprediction. In such a case, with a probability of $1-\delta$, the pseudo label y' is set to the predicted label $\hat{y}$; and with a probability of δ, the original label y is retained as the pseudo label y'. Conversely, if the predicted label $\hat{y}$ matches the original label y, indicating a correct prediction, then with a probability of $1-\delta$, the pseudo label y' is set to the second most confident label $\hat{y}'$; and with a probability of δ, the original label y is retained as the pseudo label y'. Formally, this assignment of the pseudo label y' can be expressed as:

$$y' := \begin{cases} y, & \text{if } \alpha < \delta; \\ \hat{y}', & \text{if } \hat{y} = y \wedge \alpha \geq \delta; \\ \hat{y}, & \text{if } \hat{y} \neq y \wedge \alpha \geq \delta, \end{cases}$$

where $\alpha \sim \text{Uniform}(0,1)$ is a random variable. By choosing a smaller δ, we enforce that more pseudo-labels differ from the original labels. To let the pre-trained model randomly guess a to-be-forgotten example, δ should be set to $1/C$, where C represents the number

Algorithm 3 δ-Targeted$^{\gamma}$ Label Noise Injection For Machine Unlearning

procedure UNLEARN($\boldsymbol{D}_f, \hat{\theta}, \eta, \delta$)
  $\boldsymbol{D}'_f \leftarrow \emptyset$ {Initialize an empty set}
  for each $(\boldsymbol{x}, y) \in \boldsymbol{D}_f$ **do** {Iterate over all examples in the to-be-forgotten examples}
    $\hat{y} \leftarrow \arg\max_i \sigma \circ f_{\hat{\theta}}(\boldsymbol{x})$ {The model's predicted label}
    $\hat{y}' \leftarrow \arg\max_{i \neq \hat{y}} \sigma \circ f_{\hat{\theta}}(\boldsymbol{x})$ {The model's second confident label}
    $\alpha \sim U(0,1)$ {Draw a scalar from the uniform distribution.}
    if $\hat{y} \neq y \wedge \alpha < \delta$ **then**
      $y' \leftarrow y$ {Retain the original label}
    else if $\hat{y} \neq y \wedge \alpha \geq \delta$ **then**
      $y' \leftarrow \hat{y}$ {Let the pseudo label be the model's predicted label.}
    else if $\hat{y} = y \wedge \alpha < \delta$ **then**
      $y' \leftarrow y$ {Retain the original label}
    else
      $y' \leftarrow \hat{y}'$ {Let the pseudo label be the model's second confident label.}
    end if
    $\boldsymbol{D}'_f \leftarrow \boldsymbol{D}'_f \bigcup \{(\boldsymbol{x}, y')\}$ {Add new example $(\boldsymbol{x}, y')$ to $\boldsymbol{D}'_f$}
  end for
  $\hat{\theta}'' \leftarrow \hat{\theta}$ {Initialize the parameter needed to be fine-tuned}
  while $L'(\hat{\theta}'', \boldsymbol{D}'_f) \geq \epsilon$ **do**
    Compute $\nabla L'(\hat{\theta}'', \boldsymbol{D}'_f)$ using Eq. (6.1) {Compute gradient of modified loss}
    $\hat{\theta}'' \leftarrow \hat{\theta}'' - \eta \nabla L'(\hat{\theta}'', \boldsymbol{D}'_f)$ {Update the model's parameters}
  end while
  return $\hat{\theta}''$ {Return the updated model's parameters}
end procedure=0

of unique labels.

6.3.2 Avoid Overfitting to Label Noise with a Loss Exponent λ

After injecting label noise into the to-be-forgotten examples. A set of new to-be-forgotten examples $\boldsymbol{D}'_f = \{\boldsymbol{x}_i, y'_i\}_{i=1}^{m}$ are obtained[1], where m is the size of example to be forgotten. The unlearning process aims to minimise the loss induced by the pseudo labels while preserving the accuracy on the remaining sample $\boldsymbol{D}_r$ by only the sample $\boldsymbol{D}_f$. Note that since the pseudo labels are mostly different from the original labels, small losses

[1]Note that in the rest of the Chapter when there is no confusion, we do not distinguish to-be-forgotten examples with original labels and pseudo label.

with respect to the pseudo labels imply large losses with respect to the original labels. Examples with large loss values mean that they have not been fitted by the model and are thus unlearned.

However, we notice that minimising the loss natively might lead the model to fit the pseudo labels with unnecessarily high confidence, which is not our primary objective. Our goal is to make the model mispredict the original labels of these instances using pseudo labels, but without letting the model to overfit these pseudo labels with high confidence. This distinction is crucial. If the model already predicts a pseudo label for a to-be-forgotten example correctly, there's no need to adjust its parameters further to increase the confidence of this prediction. Instead, during the unlearning procedure, the model updating should focus on the pseudo labels that it cannot predict correctly. By doing so, we can avoid unnecessary parameter adjustments, which could lead to overfit the to-be-forgotten examples and adversely affect the model's performance.

A natural approach is to dynamically remove examples from the training set once their pseudo labels can be accurately predicted during the unlearning process. This ensures that these examples are not further learned. The model should then concentrate on learning the pseudo labels of the remaining to-be-forgotten examples that it cannot predict accurately. However, we found that this should not work well in practice. The primary issue is that learning the pseudo labels for the remaining examples alters the model's parameters. After these changes, the model may not remember the pseudo labels of previously removed examples from the to-be-forgotten examples. Therefore, instead of removing an example from the training set once its pseudo label can be accurately predicted, a more effective method is to reduce the example's contribution to the optimizing objective if the example already exhibits a small loss.

Motivated by this, we propose using an exponent γ to control the loss contribution of to-be-forgotten examples during the learning process. If the model already aligns well

with the pseudo label of a to-be-forgotten example, instead of removing this example from the training set, we reduce its contribution to the loss with an exponent γ, i.e., ℓ^γ. Note that if $\gamma > 1$, when ℓ is close to zero, ℓ^γ will become smaller than ℓ; while when ℓ is larger than one, ℓ^γ will become larger than ℓ. If the model predicts the pseudo label of a to-be-forgotten example, the corresponding loss value should be close to zero. If the model struggles to predict the pseudo label of a to-be-forgotten example, that example will have a larger loss compared with other examples whose labels are correctly predicted, encouraging the model to focus more on learning such an unfitted example. This method automatically adjusts the influence of each to-be-forgotten example based on its prediction accuracy. By decreasing the loss contribution from well-predicted examples and increasing it for poorly predicted ones during the unlearning process, this method ensures more efficient and targeted unlearning.

Formally, we exponentiate cross-entropy loss function $\ell(\sigma \circ f_\theta(\boldsymbol{x}), \boldsymbol{y}')$ by a exponent λ, where $\boldsymbol{y}'$ is the one-hot encoding form of pseudo label y'. This implicitly introduces a dynamic weight associated with each example. The objective function is defined as:

$$\arg\min_\theta L'(\theta, \boldsymbol{D}'_f) = \arg\min_\theta \frac{1}{m} \sum_{(\boldsymbol{x},\boldsymbol{y}')\in \boldsymbol{D}'_f} \frac{\ell(\sigma(f_\theta)(\boldsymbol{x}), \boldsymbol{y}')^\lambda}{m}. \tag{6.1}$$

By raising the loss to the power of λ, we dynamically adjust the contribution of each instance to the overall objective. This method ensures that the model concentrates on learning the pseudo-labels it hasn't yet mastered, without over-adjusting parameters for those it already predicts correctly. By preventing overfitting to the pseudo-labels, we avoid unnecessary parameter changes that could degrade performance on the remaining data. The gradient update during fine-tuning is then

$$\theta \leftarrow \theta - \eta \nabla L'(\theta, \boldsymbol{D}'_f), \tag{6.2}$$

where η is the learning rate. By combining the above with the targeted label noise injection, our unlearning strategy effectively reduces the risk of over-adjusting the model

parameters during the fine-tuning process. We name our method δ-Targeted$^{\gamma}$ Sample Unlearning, the pseudo-code is illustrated in Algorithm 3.

6.4 Theoretical Analysis

In this section, we provide a theoretical analysis demonstrating how our method concentrates parameter updates on the specific subset of parameters associated with the to-be-forgotten examples. This targeted adjustment effectively modifies the necessary parameters while reducing the impact on unrelated parts of the model. We compare our method with sample unlearning by applying gradient ascent on original labels. Note that gradient ascent is commonly employed by existing methods (e.g., (Golatkar et al., 2020; Tarun et al., 2023; Li et al., 2023a)). In experiments, we show that benefiting from the concentrated parameter updates, our method leads to smaller changes in parameters compared with all baselines on different neural network structures. As a consequence, the smaller change in parameter further leads to the small performance degradation

We formally introduce Concentrated Parameter Updates in the following definition.

Definition 6.4.1 (Concentrated Parameter Updates)**.** *Let θ denote the original parameter set of a pre-trained model, and let θ_1 and θ_2 be two distinct sets of updated parameters. Define the changes in parameters for θ_1 and θ_2 relative to θ as $\Delta\theta_1 = \theta_1 - \theta$ and $\Delta\theta_2 = \theta_2 - \theta$, respectively. We say that θ_2 represents a more concentrated parameter update than θ_1 if*

1. $\|\Delta\theta_1\|_1 = \|\Delta\theta_2\|_1$*, and*
2. $\Delta\theta_2$ *exhibits larger changes for certain parameters and smaller influence for others, compared to a more uniformly distributed change in* $\Delta\theta_1$.

Note that the above definition restricts the total magnitude of change $\|\Delta\theta_1\|_1 = \|\Delta\theta_2\|_1$ across two sets of parameters. It ensures that comparisons are not biased by the scale of

the changes, thereby facilitating a fair comparison of parameter updates based on their distribution of changes.

Given the non-convex optimisation landscape of neural networks and their inherent non-linearity, a fully transparent statistical analysis comparing the effects of gradient ascent and descent to all parameters is ambitious. Empirically, we have found that changes in the last fully connected layer serve as an effective surrogate for monitoring changes in the model's parameters. Specifically, let denote ϕ as the parameter of the last fully connected layer, in Section 6.5.1, we demonstrate that variations in ϕ exhibit a strong dependence with changes spanning all parameters[2]. Hence, we narrow our analysis to examine the impact of the unlearning process on the last fully connected layer's parameters.

In the following theorem, we show that after unlearning, our Targeted Sample Unlearning yields a more concentrated parameter update compared to the application of gradient ascent.

Theorem 6.4.1 (Parameter Updates for δ-Targeted$^{\lambda}$)**.** *Consider a pre-trained neural network model with parameters $\hat{\theta} = \hat{\psi}, \hat{\phi}$, where $\hat{\psi}$ represents the parameter of all layers except the last fully connected layer, $\hat{\phi}$ represents the parameter of the last fully connected layer, C is the number of classes, and M is the dimension of the feature representation. Let $\boldsymbol{o} = g_{\hat{\psi}}(\boldsymbol{x}) \in \mathbb{R}^M$ represent the activation before the last linear layer for an input $\boldsymbol{x}$, with all components $\boldsymbol{o}_i > 0$ (e.g., after ReLU activation). The logits are given by $f_{\hat{\theta}}(\boldsymbol{x}) = h_{\hat{\phi}}(\boldsymbol{o}) = \hat{\phi}\boldsymbol{o}$. We compare two updated parameter sets ϕ'' and ϕ' obtained after unlearning a to-be-forgotten example $(\boldsymbol{x}, y)$:*

*1. Our δ-**Targeted**$^{\lambda}$ **Method Update (TD) with Pseudo-Label** y':*

$$\phi'' = \hat{\phi} - \eta \nabla_{\hat{\phi}} \ell(\sigma(h_{\hat{\phi}}(\boldsymbol{o})), \boldsymbol{y'}) = \hat{\phi} - \eta(\sigma(h_{\hat{\phi}}(\boldsymbol{o})) - \boldsymbol{y'})\boldsymbol{o}^{\top};$$

[2]An intuitive explanation for this phenomenon is that changes to the last fully connected layer's parameters get amplified and propagated to former layers due to the chain rule employed during back-propagation.

2. ***Gradient Ascent Update (GA) with Original Label*** y*:*

$$\phi' = \hat{\phi} + \eta \nabla_{\hat{\phi}} \ell(\sigma(h_{\hat{\phi}}(\boldsymbol{o})), \boldsymbol{y}) = \hat{\phi} + \eta'(\sigma(h_{\hat{\phi}}(\boldsymbol{o})) - \boldsymbol{y})\boldsymbol{o}^\top,$$

where σ *is the softmax function,* ℓ *is the cross-entropy loss,* $\boldsymbol{y}$ *is the one-hot encoding of the original label,* $\boldsymbol{y'}$ *is the one-hot encoding of the pseudo-label assigned by our method, and* η *and* η' *are the learning rates. According to Definition 6.4.1, the parameter updates in* ϕ'' *are more concentrated compared to those in* ϕ'*. Specifically, the updates in* ϕ'' *are concentrated on the weights associated with the pseudo-label* y' *and the original label* y*, while the updates in* ϕ' *are distributed across all classes.*

Proof Intuition. In gradient ascent, the update direction is the gradient of the loss with respect to the original label $\boldsymbol{y}$. This gradient changes all parameters associated with not only the true class y but also all other classes, leading to more uniformly distributed parameter changes. In our method, we perform gradient descent on the loss with respect to the pseudo-label $\boldsymbol{y'}$, which is the second most confident prediction. The gradient primarily affects the parameters associated with the pseudo-label y' and the activations corresponding to $\boldsymbol{o}$. Since y' is chosen to be the class the model is already somewhat confident about, the parameter changes are concentrated on the weights connecting $\boldsymbol{o}$ to y', leading to more focused parameter updates. Therefore, under the condition that the total magnitude of changes is equal, our method results in parameter changes that are more concentrated on a subset of parameters, which satisfies the concentrated parameter update in Definition 6.4.1.

Theorem 6.4.1 provides a theoretical justification for the δ-Targeted$^\lambda$ method. It is important to note that for our theoretical analysis, we set both $\lambda = 1$ and $\delta = 1$. The parameter δ controls the privacy level. Setting $\delta = 1$ ensures that the model mispredicts labels for all to-be-forgotten examples after fine-tuning. We chose $\delta = 1$ specifically to analyse how parameter changes induced by the δ-Targeted$^\lambda$ method compare to the method induced by a gradient ascent when a model is intensively fine-tuned to force

misprediction of an example. Additionally, we employ a first-order cross-entropy loss. By increasing the order of the cross-entropy loss (i.e., setting $\lambda > 1$), the change to the model should be more focused. This is because higher values of λ emphasise larger losses, causing the network to concentrate more on examples with large training losses. Empirically, results for $\lambda = 1$ and $\lambda = 3$ are presented in Section 6.5.1. The results are consistent where $\lambda = 3$ performs better in most cases when the size of to-be-forgotten examples is large.

6.4.1 Detailed Proof

We define some notations used in the proof first. Let $(\boldsymbol{x}, y)$ by an example. Let $\boldsymbol{o}$ represent the activation just before the last layer of the pre-trained model. For the label y, its corresponding one-hot encoding is denoted by $\boldsymbol{y}$. The estimated class posteriors $\boldsymbol{p} = \sigma \circ h_{\hat{\phi}}(\boldsymbol{o})$ are defined by the output of the softmax function applied to the linear transformation. It can be further expanded to $\boldsymbol{p} = \sigma(\hat{\phi}\boldsymbol{o}) = \sigma(\boldsymbol{z})$, where $\hat{\phi} \in \mathbb{R}^{C \times M}$ represents the parameter of the last fully connected layer, $\boldsymbol{z} = \hat{\phi}\boldsymbol{o}$ represents the input to the softmax function, and σ is the softmax function.

Firstly, we derive the gradient of cross-entropy loss with respect to the last layer parameter ϕ. The cross-entropy loss function $\ell(\boldsymbol{p}, \boldsymbol{y})$ is defined as:

$$\ell(\boldsymbol{p}, \boldsymbol{y}) = -\sum_{k=1}^{C} \boldsymbol{y}_k \cdot \log(\boldsymbol{p}_k). \tag{6.3}$$

The gradient of ℓ with respect to the predicted class-posterior probability $\boldsymbol{p}_k = P(Y = k|X = \boldsymbol{x})$ can be calculated as:

$$\frac{\partial \ell}{\partial \boldsymbol{p}_k} = -\frac{\boldsymbol{y}_k}{\boldsymbol{p}_k}. \tag{6.4}$$

Consider applying softmax function $\sigma(z)_i$ to calculate i-th class posterior $\boldsymbol{p}_i = \sigma(z)_i = \frac{e^{\boldsymbol{z}_i}}{\sum_{j=1}^{C} e^{\boldsymbol{z}_j}}$.

Case when $i = k$, the derivative $\frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{z}_k} = \frac{\partial \boldsymbol{p}_k}{\partial \boldsymbol{z}_k}$ is

$$\begin{aligned}\frac{\partial \boldsymbol{p}_k}{\partial \boldsymbol{z}_k} &= \frac{\sum_{j=1}^{C} e^{\boldsymbol{z}_j} \cdot e^{\boldsymbol{z}_k} - e^{\boldsymbol{z}_k} \cdot e^{\boldsymbol{z}_k}}{(\sum_{j=1}^{C} e^{\boldsymbol{z}_j})^2} \\ &= \frac{e^{\boldsymbol{z}_k}}{\sum_{j=1}^{C} e^{\boldsymbol{z}_j}} \cdot (1 - \frac{e^{\boldsymbol{z}_k}}{\sum_{j=1}^{C} e^{\boldsymbol{z}_j}}) \\ &= \boldsymbol{p}_k(1 - \boldsymbol{p}_k).\end{aligned}$$

Case when $i \neq k$, the derivative $\frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{z}_k}$ is

$$\begin{aligned}\frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{z}_k} &= \frac{0 - e^{\boldsymbol{z}_k} \cdot e^{\boldsymbol{z}_i}}{(\sum_{j=1}^{C} e^{\boldsymbol{z}_j})^2} \\ &= -\frac{e^{\boldsymbol{z}_k} \cdot e^{\boldsymbol{z}_i}}{(\sum_{j=1}^{C} e^{\boldsymbol{z}_j})^2} \\ &= -\boldsymbol{p}_k \boldsymbol{p}_i.\end{aligned}$$

Applying the chain rule, the gradient of ℓ with respect to $\boldsymbol{z}_k$ becomes:

$$\frac{\partial \ell}{\partial \boldsymbol{z}_k} = \sum_i \frac{\partial \ell}{\partial \boldsymbol{p}_i} \frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{z}_k} = -\boldsymbol{y}_k(1 - \boldsymbol{p}_k) + \sum_{j \neq k} \boldsymbol{y}_j \boldsymbol{p}_k. \tag{6.5}$$

Note that $\boldsymbol{y}$ is the one-hot encoding form of y, $\boldsymbol{y}_k$ can only include a value of either 0 or 1. The above equation simplifies to $\boldsymbol{p}_k - 1$ when $y = k$ and $\boldsymbol{p}_k$ when $y \neq k$. In matrix form, we can rewrite it as:

$$\frac{\partial \ell}{\partial \boldsymbol{z}} = \boldsymbol{p} - \boldsymbol{y}. \tag{6.6}$$

Furthermore, the derivative of logits $\boldsymbol{z}$ with respect to the last-layer parameter ϕ is $\frac{\partial \boldsymbol{z}}{\partial \phi} = \boldsymbol{o}^T$. Then, by applying the chain rule again, the gradient of the loss ℓ with respect to the parameter ϕ is given by:

$$\frac{\partial \ell}{\partial \phi} = \frac{\partial \ell}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \phi} = (\boldsymbol{p} - \boldsymbol{y})\boldsymbol{o}^T. \tag{6.7}$$

To unlearn the example $(\boldsymbol{x}, y)$, we update ϕ to ϕ' using gradient ascent: $\phi' = \phi + \eta' \nabla_{\hat{\phi}} \ell(h_{\hat{\phi}}(\boldsymbol{o}), \boldsymbol{y})$, where $\eta' \in R$ is the learning rate, and $\eta' > 0$. The updated elements of ϕ' are defined as:

$$\phi'_y = \phi_y + \eta'(\boldsymbol{p}_y - 1)\boldsymbol{o}^T,$$
$$\phi'_k = \phi_k + \eta'(\boldsymbol{p}_k)\boldsymbol{o}^T, \text{ for } k \neq y.$$

Let $\Delta \boldsymbol{p} = \boldsymbol{p} - \boldsymbol{y}$, we write the above equations by using gradient ascent in matrix form

$$\phi' = \phi + \eta'(\boldsymbol{p} - \boldsymbol{y})\boldsymbol{o}^T$$
$$= \phi + \eta' \Delta \boldsymbol{p} \boldsymbol{o}^T.$$

We also consider an update of ϕ to ϕ'' using gradient descent: $\phi'' = \phi - \eta \nabla_{\hat{\phi}} L(h_{\hat{\phi}}(\boldsymbol{o}), y')$, where $\eta \in R$ is the learning rate, and $\eta > 0$. The updated elements of ϕ'' are defined as:

$$\phi''_{y'} = \phi_{y'} + \eta(1 - \boldsymbol{p}_{y'})\boldsymbol{o}^T,$$
$$\phi''_k = \phi_k + \eta \boldsymbol{p}_k \boldsymbol{o}^T,$$
$$\phi''_y = \phi_y + \eta \boldsymbol{p}_y \boldsymbol{o}^T, \text{ for } k \neq y, y'.$$

Let $\Delta \boldsymbol{p}' = \boldsymbol{y}' - \boldsymbol{p}$, we write the above equations by using gradient descent in matrix form

$$\phi'' = \phi + \eta(\boldsymbol{y}' - \boldsymbol{p})\boldsymbol{o}^T$$
$$= \phi + \eta \Delta \boldsymbol{p}' \boldsymbol{o}^T.$$

The L_1-norms of $\Delta\phi'$ and $\Delta\phi''$ are given by:

$$||\Delta\phi'||_1 = \eta' ||\Delta \boldsymbol{p} \boldsymbol{o}^T||_1,$$
$$||\Delta\phi''||_1 = \eta ||\Delta \boldsymbol{p}' \boldsymbol{o}^T||_1.$$

We now compare the effect of gradient ascent and the gradient descent when $\Delta\phi'$ and $\Delta\phi''$ have the same L_1 norm. Recall $\Delta\boldsymbol{p} = \boldsymbol{p} - \boldsymbol{y}$ and $\Delta\boldsymbol{p}' = \boldsymbol{y}' - \boldsymbol{p}$, we can derive the following points:

- For $i \neq y$ and $i \neq y'$, we have $|\Delta\boldsymbol{p}'_i| = |-\boldsymbol{p}_i|$ and $|\Delta\boldsymbol{p}_i| = |\boldsymbol{p}_i|$.
- When $i = y$, we have $|\Delta\boldsymbol{p}'_i| = |-\boldsymbol{p}_y|$ and $|\Delta\boldsymbol{p}_i| = |\boldsymbol{p}_y - 1|$.
- When $i = y'$, we have $|\Delta\boldsymbol{p}'_i| = |1 - \boldsymbol{p}_{y'}|$ and $|\Delta\boldsymbol{p}_i| = |-\boldsymbol{p}_{y'}|$.

Therefore, the L_1-norms of $\Delta\boldsymbol{p}$ and $\Delta\boldsymbol{p}'$ are:

$$||\Delta\boldsymbol{p}||_1 = |\boldsymbol{p}_{y'}| + |1 - \boldsymbol{p}_y| + \sum_{i \neq y, y'} |\boldsymbol{p}_i| = 2 - 2\boldsymbol{p}_y,$$
$$||\Delta\boldsymbol{p}'||_1 = |1 - \boldsymbol{p}_{y'}| + |\boldsymbol{p}_y| + \sum_{i \neq y, y'} |-\boldsymbol{p}_i| = 2 - 2\boldsymbol{p}_{y'}.$$

Given the fact that y is most confident label, then $\boldsymbol{p}_y > \boldsymbol{p}_{y'}$, we then have:

$$\begin{aligned} ||\Delta\boldsymbol{p}||_1 - ||\Delta\boldsymbol{p}'||_1 =& 2 - 2\boldsymbol{p}_y - (2 - 2\boldsymbol{p}_{y'}) \\ =& -2\boldsymbol{p}_y + 2\boldsymbol{p}_{y'} < 0. \end{aligned}$$

Let $\eta' = \frac{1}{2-2\boldsymbol{p}_y}$ and $\eta = \frac{1}{2-2\boldsymbol{p}_{y'}}$ to ensure $\Delta\phi'$ and $\Delta\phi''$ have the same L_1 norm. Then, we can rewrite the update rules for ϕ' and ϕ'' as follows:

$$\phi'_{*,j} = \phi_{*,j} + \eta'(\boldsymbol{p} - \boldsymbol{y})\boldsymbol{o}_j = \phi_{*,j} + \frac{1}{2 - 2\boldsymbol{p}_y}(\boldsymbol{p} - \boldsymbol{y})\boldsymbol{o}_j,$$
$$\phi''_{*,j} = \phi_{*,j} + \eta(\boldsymbol{y}' - \boldsymbol{p})\boldsymbol{o}_j = \phi_{*,j} + \frac{1}{2 - 2\boldsymbol{p}_{y'}}(\boldsymbol{y}' - \boldsymbol{p})\boldsymbol{o}_j,$$

where $\boldsymbol{M}_{*,j}$ denote the j-th column in a matrix $\boldsymbol{M}$.

Now let us discuss the following two cases to compare the change of parameters:

1). For all $i \neq y, y'$, we have:

$$\phi'_{ij} = \phi_{ij} + \frac{1}{2-2\boldsymbol{p}_y}\boldsymbol{p}_i\boldsymbol{o}_j, \tag{6.8}$$

$$\phi''_{ij} = \phi_{ij} - \frac{1}{2-2\boldsymbol{p}_{y'}}\boldsymbol{p}_i\boldsymbol{o}_j. \tag{6.9}$$

Given that $\boldsymbol{p}_{y'} < \boldsymbol{p}_y$, it follows that $|\phi''_{ij}| - |\phi'_{ij}| < 0$. Thus, when the L_1 norm of $\Delta\phi'$ and $\Delta\phi''$ are the same, the gradient ascent method leads to more changes in the parameters for predicting $\boldsymbol{p}_i$ for $i \neq y$ and $i \neq y'$ compared to gradient descent.

2). For $i = y$, we have:

$$\phi'_{ij} = \phi_{ij} + \frac{1}{2-2\boldsymbol{p}_y}(\boldsymbol{p}_y - 1)\boldsymbol{o}_j, \tag{6.10}$$

$$\phi''_{ij} = \phi_{ij} - \frac{1}{2-2\boldsymbol{p}_{y'}}\boldsymbol{p}_y\boldsymbol{o}_j. \tag{6.11}$$

Observing that $|\phi''_{ij}| - |\phi'_{ij}| = \frac{-1+\boldsymbol{p}_{y'}+\boldsymbol{p}_y}{2(1-\boldsymbol{p}_{y'})} < 0$ when having more than two classes. Then, when the L_1 norms of $\Delta\phi'$ and $\Delta\phi''$ are identical, the gradient ascent method leads to more changes in the parameters for predicting $\boldsymbol{p}_y$ than gradient descent.

Combining the findings for both cases, $|\phi''_{ij}| - |\phi'_{ij}| < 0$. As L_1 norms of $\Delta\phi'$ and $\Delta\phi''$ are identical, we conclude that only for $i = y'$, the value of $|\phi''_{ij}| - |\phi'_{ij}| > 0$. This means our method only leads to large changes in the parameters for predicting pseudo-label y'.

When more than two classes, our method results in parameter updates that are more concentrated on the weights associated with the pseudo-label y', while gradient ascent distributes updates more uniformly across all classes, which completes the proof.

6.5 Experiments

Datasets and Baselines The evaluation of our unlearning methods was conducted using four different benchmark image classification datasets: CIFAR-10, CIFAR-100

(Krizhevsky et al., 2009), Fashion-MNIST (Xiao et al., 2017), and Tiny-ImageNet (Le and Yang, 2015). CIFAR-10 contains 10 classes with 50,000 training images and 10,000 test images. CIFAR-100 includes 100 classes with 50,000 training images and 10,000 test images. Fashion-MNIST has 10 classes with 60,000 training images and 10,000 test images. Tiny-ImageNet contains 200 classes with 100,000 training images and 10,000 test images.

We utilized ResNet-18, ResNet-50 (He et al., 2016) and vision transformer (ViT) architectures (Dosovitskiy et al., 2020) for our experiments. We established a comparison of our proposed methods against different baselines. These include "*BEFORE*", which represents the state of the model before unlearning; "NegGrad", which involves fine-tuning models using gradient ascent (Kurmanji et al., 2024); "RandL", which misleads the model through random label perturbation (Huang et al., 2024b); "Bad Teacher", which uses competent and incompetent teachers in a student-teacher framework to induce forgetfulness (Chundawat et al., 2023); "Amnesiac", which undoes the parameter updates from only the batches containing sensitive data (Graves et al., 2021). Our proposed methods are δ-Targeted^1, which sets the $\lambda = 1$ and δ-Targeted^3, which sets $\lambda = 3$. Note that for "Bad Teacher" and "Amnesiac", some remaining examples need to be accessed. In our setting, we provide them with 300 remaining examples. For all other methods, no remaining examples are provided.

Experiment Settings For each dataset, a random subset of images from the training dataset was selected to form the unlearning sample, denoted as $\boldsymbol{D}_f$. The size of this sample varied in the set $\{4, 16, 64, 128, 256, 512, 1024\}$. The unlearning process was carried out using an Adam optimiser. The parameters of the Adam optimizer were the same for all experiments, with a learning rate of $1e-4$, a weight decay of $1e-5$. To ensure statistical significance, all experiments were conducted five times, each with different random initialisations for each setting. The standard deviation of the results from each

five trials has also been reported.

Note that to make the pre-trained model randomly guess a to-be-forgotten example, δ should be set to $1/C$, where C is the number of classes. In our experiments, we test our method under the most challenging conditions, i.e., $\delta = 0$. This requires the model to misclassify all to-be-forgotten examples by learning from the label noise while preserving performance on the remaining examples.

6.5.1 Magnitude of Parameter Changes for Different Methods

In Section 6.4, we analysed that our method results in concentrated parameter updates for the last-layer parameters. We hypothesise that concentrated updates in the last-layer parameters should lead to smaller changes in all parameters. We validate this on different neural network architectures, datasets, and baselines.

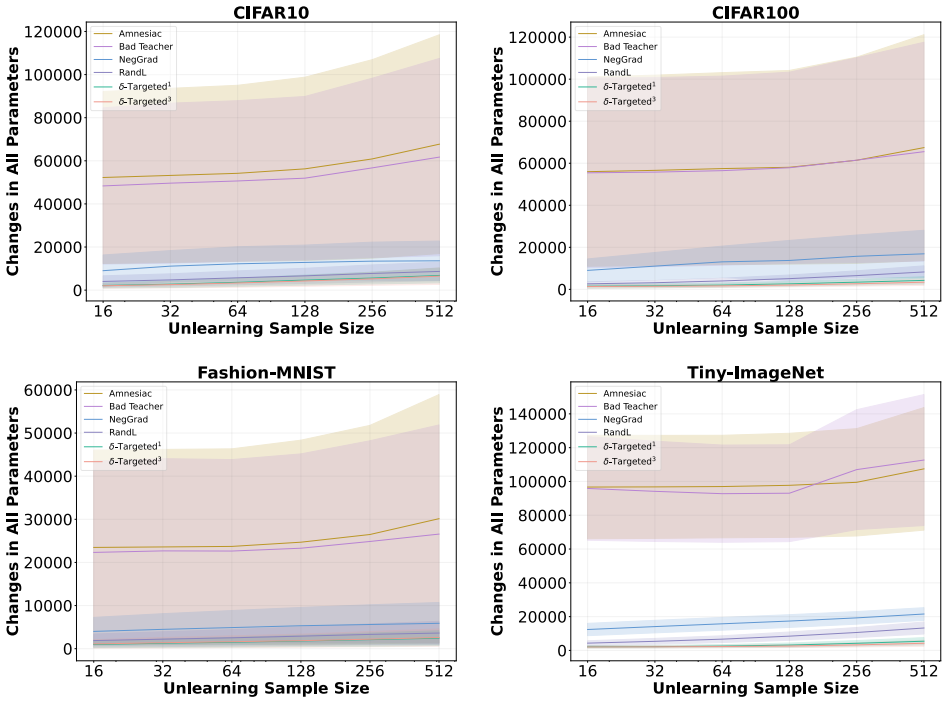

Figure 6.1: Parameter changes for different numbers of to-be-forgotten examples across different datasets. The figure includes three neural network architectures: ResNet-18, ResNet-50, and ViT. The changes are quantified by the L_1-norm. Our method results in the smallest changes to the models' parameters across the datasets.

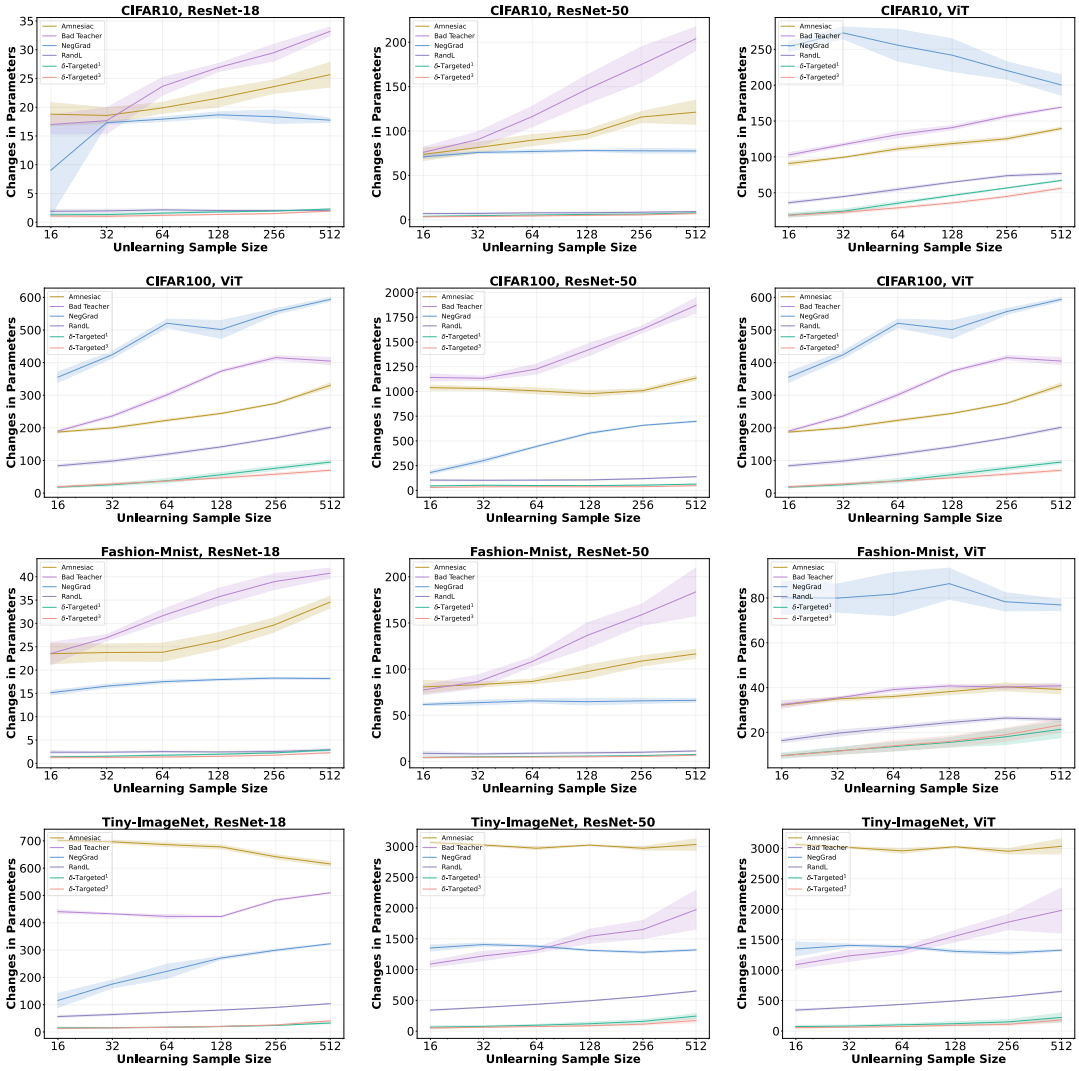

Figure 6.2: The magnitude of parameter changes across the model, quantified by the L_1-norm. We use $\Delta\theta$ to denote the change in all parameters and $\Delta\phi$ to denote the change in the last layer's parameters. Our methods lead to the smallest change to the models' parameters on different datasets.

6.5.2 Changes in All Parameters for Different Methods.

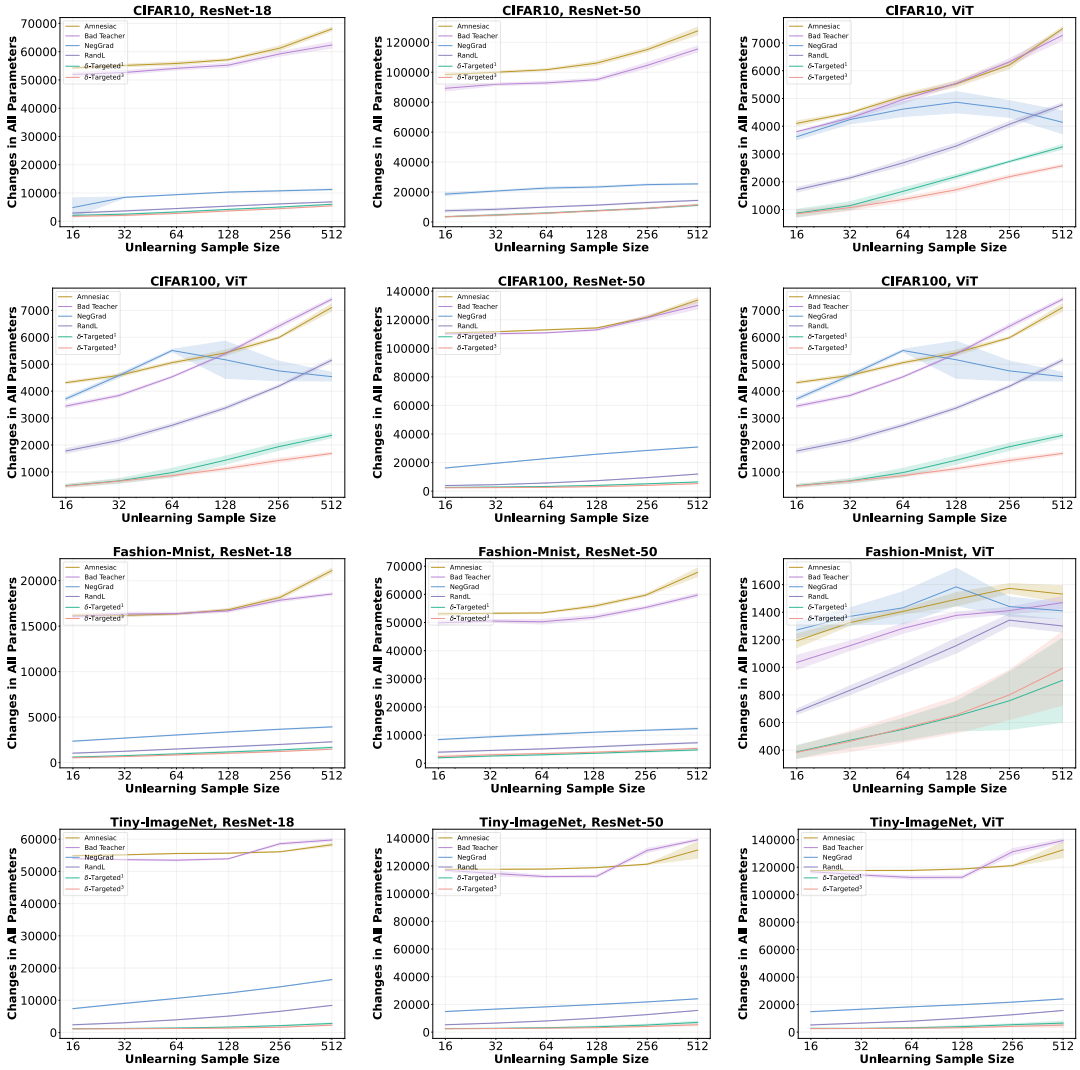

Figure 6.3: The parameter changes across different neural network structures. The changes are quantified by the L_1-norm. Our method leads to the smallest change to models' parameters on different datasets and across different neural network structures.

In Figures (original: Fig.) 6.1 - 6.3, we measure and compare the parameter changes introduced by different unlearning methods, quantified using the L_1-norm. The results confirm that our proposed method consistently induces smaller parameter changes after unlearning. This empirical evidence supports our method's effectiveness in not only forgetting specific data but also in inducing only small changes to the model's parameters. Moreover, the consistent pattern of small parameter changes observed across various datasets and neural network architectures underscores the robustness of our method across different datasets and models.

6.5.3 Relations between Accuracies and Parameter Changes

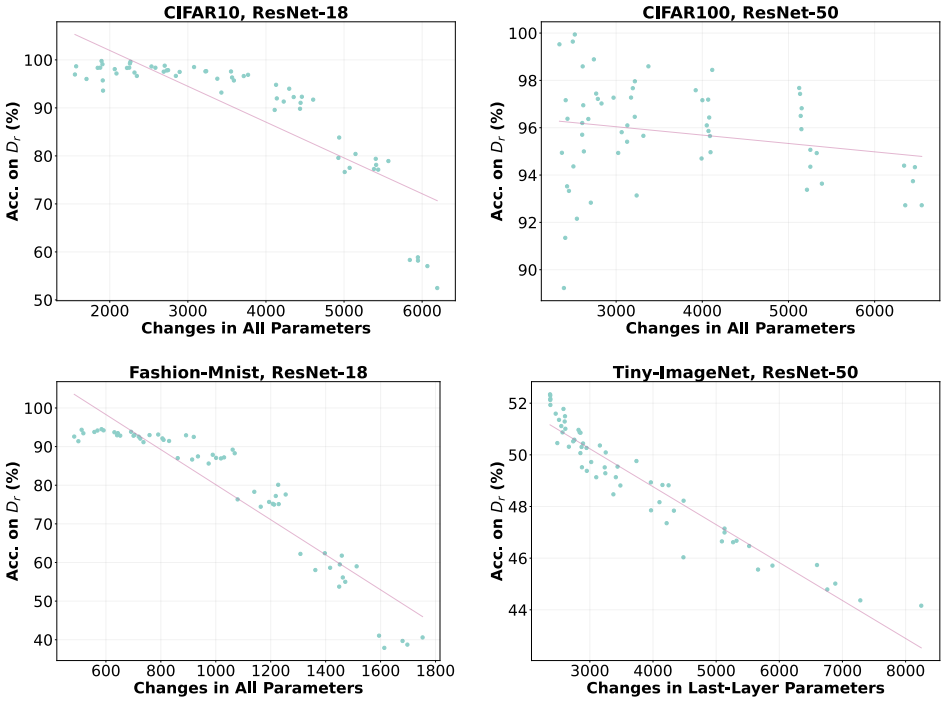

Figure 6.4: Dependence between changes in all parameters and model accuracy on remaining examples by using our method. The result shows a negative correlation. The results show a positive correlation.

In Figure 6.4, we visualise the relationship between the change in all parameters of the model and test accuracy using scatter plots. The change in the models' parameters is

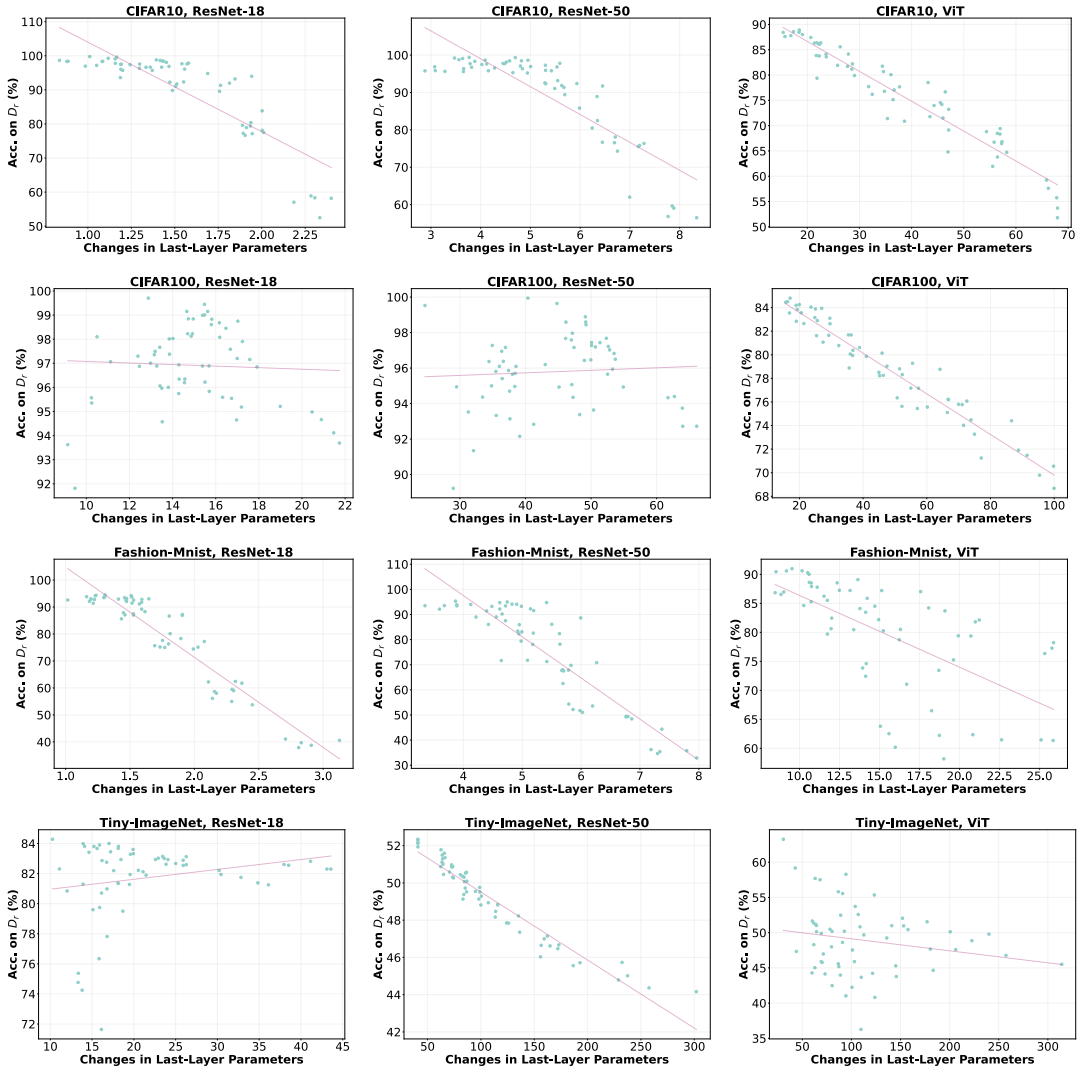

Figure 6.5: Dependence between changes in last-layer parameters and model accuracy on remaining examples by using our method. The result shows a negative correlation.

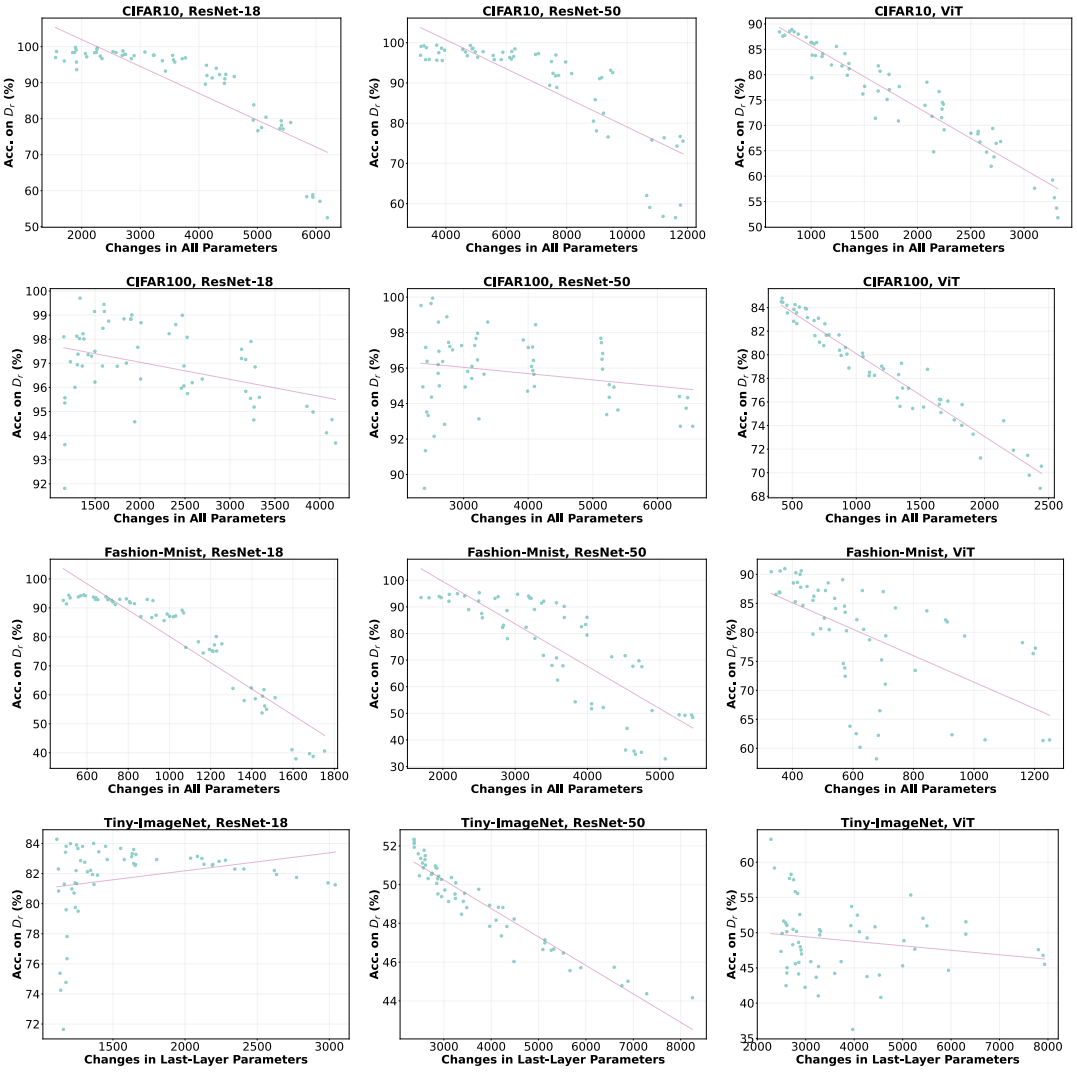

Figure 6.6: Dependence between changes in all parameters and model accuracy on remaining examples by using our method. The result shows a negative correlation.

quantified using the L_1-norm. The dependence between changes in last-layer parameters and model accuracy on remaining examples using our method is illustrated in Figure 6.5.

For each subplot, δ-Targeted^1 and δ-Targeted^3 are collectively displayed. A linear regression line obtained by regressing models' test accuracies against changes in parameters is also shown. The results indicate that there is a consistent negative dependence between the change in the models' parameters $\Delta\theta$ and test accuracy across various datasets. These findings suggest that small changes in parameters during the unlearning process contribute to maintaining test accuracy. This confirms that our method, by inducing small changes to the model's parameters, successfully maintains the performance of the model after unlearning.

Additionally, smaller models seem to be more sensitive to parameter changes, which impacts their generalisation ability. By examining Figure 6.4 and Figure 6.6, it is evident that across most datasets, the same amount of parameter change causes a larger drop in accuracy for smaller neural network models. Another interesting observation from the figure is that the rate at which accuracy decreases in response to parameter changes varies by dataset. The accuracy drop is fastest on Fashion-MNIST, followed by CIFAR-10, CIFAR-100, and Tiny-ImageNet. This trend may suggest that models trained on more complex tasks are not as vulnerable to losses in accuracy due to parameter changes.

6.5.4 Relations between Changes in Last-Layer Parameters and Changes in All Parameters

In Section 6.4, we showed that our method results in concentrated parameter updates in the last-layer parameters. We focused on parameter changes in the last layer instead of changes in all parameters of a model. We hypothesise that updates in the last-layer parameters should lead to smaller changes in all parameters. We validate this on

different neural network architectures and datasets in this subsection.

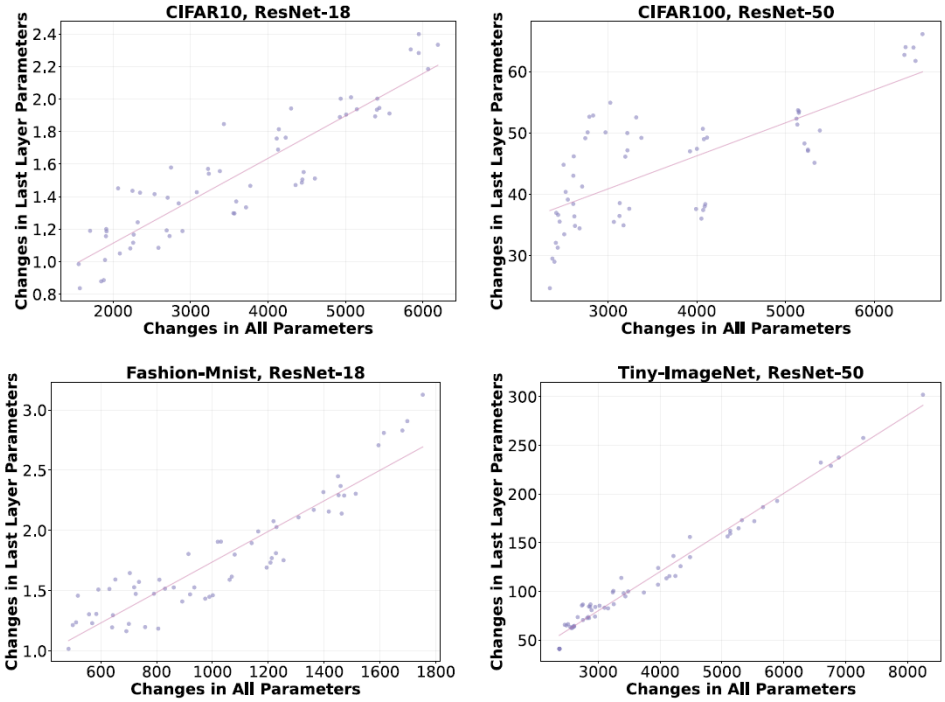

Figure 6.7: Dependence between the changes in all parameters and the changes in the last layer parameters by using our method. The magnitude of the parameter changes is quantified by the L_1-norm. The results show a positive correlation.

Figures 6.7 and 6.8 demonstrate the relationship between changes in the last-layer parameters and changes in a model's overall parameters. Each subfigure represents a dataset and neural network architecture combination. The data reveal a consistent positive correlation between changes in the last-layer parameters and changes in a model's overall parameters. This validated our guess that changes in the last layer's parameters can effectively represent changes across all parameters of a model

6.5.5 Performance on Different Datasets

Tables 6.1, 6.2, 6.3, and 6.4 present the classification accuracies on different datasets with different methods for various sizes of the unlearning sample $\boldsymbol{D}_f$. Note that the results

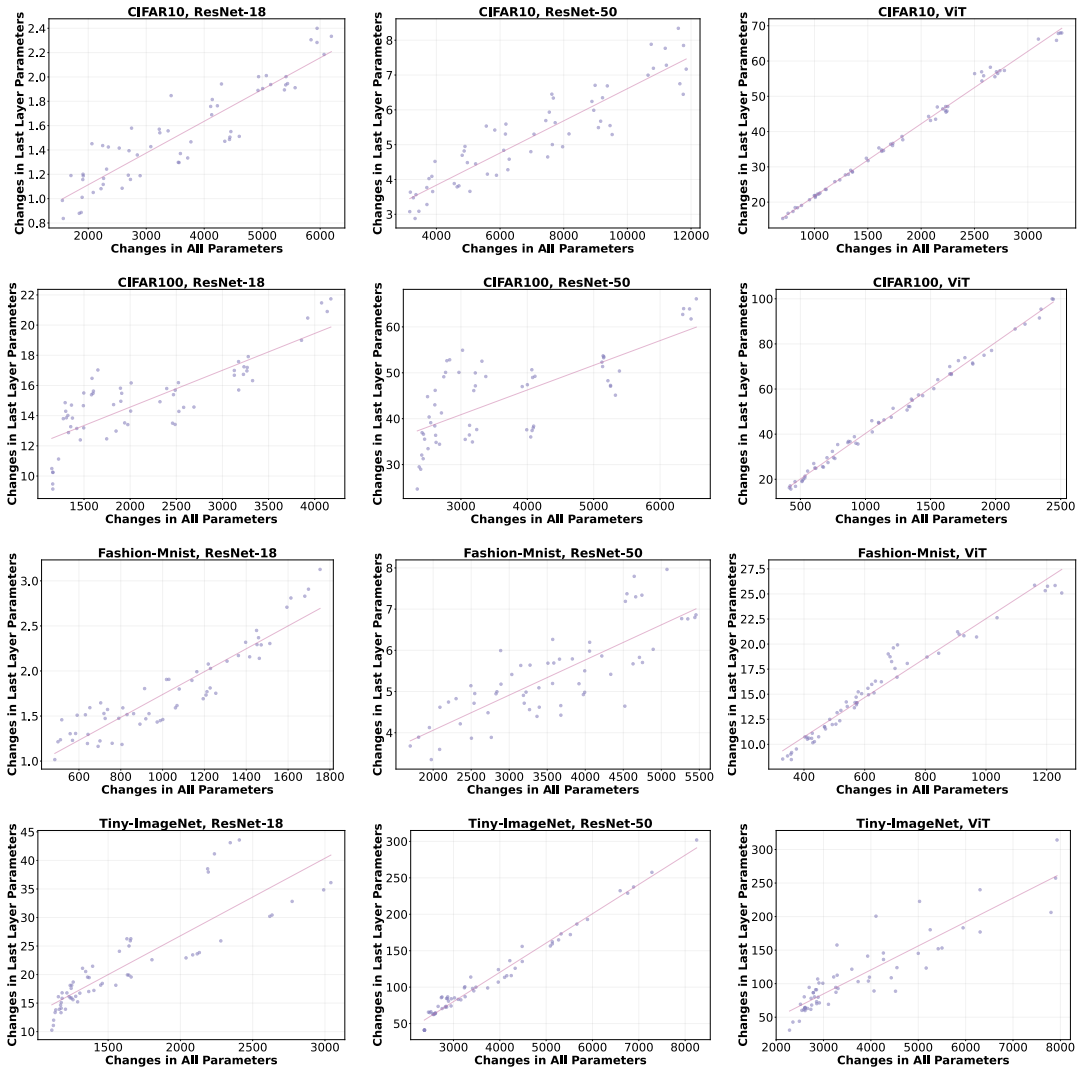

Figure 6.8: Dependence between the changes in all parameters and the changes in the last layer parameters by using our method. The magnitude of the parameter changes is quantified by the L_1-norm. The result shows a positive correlation.

Table 6.1: Means and standard deviations (percentage) of classification accuracy on CIFAR10.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	97.92 ± 4.93	98.12 ± 3.39	98.33 ± 3.15	98.39 ± 2.73	98.26 ± 2.79	97.83 ± 3.17
	NegGrad	16.25 ± 33.68	3.33 ± 4.90	2.92 ± 4.30	3.12 ± 4.60	3.62 ± 5.19	3.48 ± 4.98
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	11.25 ± 8.29	9.38 ± 4.70	11.46 ± 3.68	12.34 ± 4.64	13.18 ± 4.34	16.07 ± 7.20
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.18 ± 0.35	0.90 ± 1.40
$\boldsymbol{D_r}$	*BEFORE*	97.88 ± 3.00	97.88 ± 3.00	97.88 ± 3.00	97.88 ± 3.00	97.87 ± 3.00	97.88 ± 2.99
	NegGrad	69.62 ± 22.88	53.10 ± 24.18	32.81 ± 15.36	17.46 ± 6.98	9.29 ± 1.36	6.64 ± 2.49
	RandL	87.98 ± 6.54	84.92 ± 7.67	74.79 ± 10.21	59.97 ± 9.95	42.47 ± 9.91	28.87 ± 7.34
	Bad Teacher	78.09 ± 6.95	78.11 ± 7.25	75.75 ± 7.26	69.23 ± 10.53	56.55 ± 14.55	40.50 ± 13.56
	Amnesiac	76.83 ± 4.44	75.84 ± 4.27	73.77 ± 3.97	68.89 ± 4.39	54.21 ± 7.68	34.29 ± 6.20
	δ-Targeted[1]	**94.35 ± 6.48**	**93.73 ± 7.13**	89.34 ± 10.51	84.57 ± 10.22	75.32 ± 7.38	57.13 ± 2.69
	δ-Targeted[3]	93.24 ± 4.51	93.61 ± 5.50	**91.98 ± 7.80**	**90.55 ± 8.50**	**86.22 ± 7.98**	**73.78 ± 4.74**
$\boldsymbol{D_t}$	*BEFORE*	89.62 ± 5.62	89.62 ± 5.62	89.62 ± 5.62	89.62 ± 5.62	89.62 ± 5.62	89.62 ± 5.62
	NegGrad	64.41 ± 20.62	48.77 ± 21.49	29.71 ± 13.12	15.61 ± 5.62	8.70 ± 1.45	6.39 ± 2.58
	RandL	80.15 ± 7.11	76.86 ± 7.97	67.25 ± 9.89	53.41 ± 8.61	37.44 ± 8.49	25.10 ± 6.12
	Bad Teacher	73.53 ± 4.35	73.29 ± 5.23	71.26 ± 5.47	65.24 ± 9.19	53.46 ± 13.09	38.69 ± 12.63
	Amnesiac	72.14 ± 2.42	71.14 ± 3.34	69.16 ± 3.98	64.62 ± 4.46	50.70 ± 6.62	32.24 ± 5.38
	δ-Targeted[1]	**85.73 ± 7.47**	**85.26 ± 8.09**	81.17 ± 10.64	76.43 ± 10.00	67.36 ± 7.11	50.68 ± 2.56
	δ-Targeted[3]	84.59 ± 5.74	85.08 ± 6.58	**83.62 ± 8.55**	**82.17 ± 9.07**	**77.75 ± 8.32**	**65.93 ± 4.64**

Table 6.2: Means and standard deviations (percentage) of classification accuracy on CIFAR100.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	95.42 ± 8.06	93.96 ± 9.98	94.48 ± 7.94	94.84 ± 7.19	94.71 ± 7.38	94.92 ± 7.17
	NegGrad	0.42 ± 1.56	0.21 ± 0.78	0.21 ± 0.53	7.03 ± 24.85	0.26 ± 0.47	0.23 ± 0.43
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	1.25 ± 3.39	1.25 ± 1.91	2.08 ± 2.03	2.24 ± 2.23	3.12 ± 2.58	4.21 ± 4.37
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	95.04 ± 6.97	95.04 ± 6.97	95.04 ± 6.96	95.04 ± 6.97	95.04 ± 6.97	95.04 ± 6.97
	NegGrad	62.90 ± 19.23	59.81 ± 25.59	53.14 ± 29.77	47.59 ± 34.69	41.18 ± 29.50	33.36 ± 23.49
	RandL	82.07 ± 11.83	83.30 ± 14.32	81.64 ± 17.73	78.32 ± 22.15	73.81 ± 26.68	67.00 ± 31.73
	Bad Teacher	44.56 ± 21.53	43.94 ± 20.06	40.29 ± 17.11	31.14 ± 17.08	23.36 ± 14.10	15.56 ± 9.77
	Amnesiac	41.52 ± 16.65	40.75 ± 15.86	38.51 ± 14.30	35.55 ± 13.66	26.92 ± 14.24	16.78 ± 10.37
	δ-Targeted[1]	**93.39 ± 6.96**	**92.85 ± 7.43**	**92.00 ± 8.63**	**90.64 ± 10.10**	89.32 ± 11.06	86.20 ± 11.15
	δ-Targeted[3]	90.88 ± 5.42	91.58 ± 6.31	91.09 ± 7.33	90.28 ± 8.20	**89.70 ± 8.62**	**88.39 ± 9.12**
$\boldsymbol{D_t}$	*BEFORE*	67.07 ± 8.89	67.07 ± 8.89	67.07 ± 8.89	67.07 ± 8.89	67.07 ± 8.89	67.07 ± 8.89
	NegGrad	43.66 ± 11.78	41.15 ± 15.74	36.29 ± 18.85	32.19 ± 23.34	27.50 ± 19.32	22.38 ± 15.39
	RandL	55.43 ± 7.83	56.48 ± 9.79	55.72 ± 12.13	53.40 ± 14.77	49.57 ± 17.30	43.51 ± 19.77
	Bad Teacher	34.26 ± 12.22	33.89 ± 11.57	31.59 ± 10.02	24.47 ± 11.97	18.69 ± 10.84	12.91 ± 8.09
	Amnesiac	32.08 ± 8.73	31.51 ± 8.30	30.01 ± 7.45	27.75 ± 7.52	20.64 ± 8.64	13.03 ± 6.84
	δ-Targeted[1]	**63.24 ± 7.11**	**63.08 ± 6.99**	**62.87 ± 7.98**	**62.13 ± 8.73**	60.85 ± 9.22	57.84 ± 8.43
	δ-Targeted[3]	60.70 ± 5.37	61.68 ± 5.97	61.92 ± 6.97	61.67 ± 7.41	**60.97 ± 7.67**	**59.43 ± 7.62**

Table 6.3: Means and standard deviations (percentage) of classification accuracy on Fashion-MNIST.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	96.25 ± 5.00	95.42 ± 4.25	94.79 ± 3.16	94.95 ± 2.73	95.13 ± 2.11	95.76 ± 1.78
	NegGrad	2.92 ± 6.80	2.71 ± 6.54	3.65 ± 6.42	2.24 ± 4.33	3.36 ± 5.03	3.48 ± 4.99
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.46 ± 0.68
	Bad Teacher	7.92 ± 8.06	6.04 ± 5.41	6.77 ± 5.84	6.93 ± 5.51	8.83 ± 6.67	8.87 ± 7.05
	Amnesiac	0.00 ± 0.00	0.42 ± 1.56	2.50 ± 3.41	6.30 ± 8.82	12.37 ± 13.05	10.68 ± 10.84
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.62 ± 1.11	1.09 ± 1.91	4.56 ± 6.83	17.50 ± 24.87
	δ-Targeted[3]	11.25 ± 16.65	10.42 ± 15.21	15.62 ± 22.37	22.40 ± 31.67	25.55 ± 36.15	25.85 ± 36.57
$\boldsymbol{D_r}$	*BEFORE*	96.04 ± 1.37	96.04 ± 1.37	96.04 ± 1.37	96.04 ± 1.37	96.04 ± 1.36	96.04 ± 1.36
	NegGrad	53.98 ± 24.96	41.26 ± 22.91	30.46 ± 14.70	17.86 ± 7.91	11.69 ± 3.20	8.02 ± 2.19
	RandL	72.05 ± 17.74	64.08 ± 23.78	53.59 ± 24.06	40.22 ± 19.96	27.35 ± 14.38	18.87 ± 8.96
	Bad Teacher	79.16 ± 10.05	76.20 ± 12.20	71.15 ± 15.48	63.17 ± 17.87	49.02 ± 17.18	37.71 ± 14.54
	Amnesiac	81.51 ± 6.08	80.15 ± 5.16	74.83 ± 7.61	69.01 ± 8.52	56.34 ± 6.37	40.00 ± 3.08
	δ-Targeted[1]	91.76 ± 3.26	88.56 ± 3.31	82.73 ± 3.38	72.52 ± 3.96	57.43 ± 5.38	45.50 ± 11.91
	δ-Targeted[3]	**92.47 ± 1.68**	**91.98 ± 2.15**	**90.04 ± 2.48**	**85.04 ± 2.44**	**75.70 ± 4.92**	**61.69 ± 11.73**
$\boldsymbol{D_t}$	*BEFORE*	91.95 ± 0.33	91.95 ± 0.33	91.95 ± 0.33	91.95 ± 0.33	91.95 ± 0.33	91.95 ± 0.33
	NegGrad	51.19 ± 23.07	38.70 ± 21.18	28.15 ± 13.11	16.12 ± 6.77	10.58 ± 2.58	7.39 ± 2.37
	RandL	68.66 ± 16.13	60.82 ± 21.88	50.34 ± 22.10	37.26 ± 18.06	24.65 ± 12.58	16.47 ± 7.52
	Bad Teacher	77.94 ± 10.06	74.83 ± 12.05	69.89 ± 15.42	61.94 ± 17.84	47.99 ± 17.12	37.14 ± 14.48
	Amnesiac	80.24 ± 6.23	78.87 ± 5.31	73.62 ± 7.83	67.77 ± 8.67	55.18 ± 6.86	38.94 ± 3.54
	δ-Targeted[1]	87.99 ± 1.62	84.85 ± 2.47	79.04 ± 3.16	68.78 ± 4.71	54.04 ± 6.72	42.59 ± 13.57
	δ-Targeted[3]	**88.89 ± 0.85**	**88.34 ± 0.75**	**86.27 ± 1.27**	**81.36 ± 2.64**	**72.14 ± 6.60**	**58.54 ± 13.48**

Table 6.4: Means and standard deviations (percentage) of classification accuracy on Tiny-ImageNet.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	64.53 ± 24.87	65.17 ± 24.07	64.59 ± 23.62	65.13 ± 23.10	65.32 ± 22.02	66.52 ± 23.84
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	0.42 ± 1.56	0.21 ± 0.78	0.00 ± 0.00	0.42 ± 0.48	0.31 ± 0.36	0.42 ± 0.40
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.05
$\boldsymbol{D_r}$	*BEFORE*	71.22 ± 15.44	71.22 ± 15.44	71.22 ± 15.44	71.22 ± 15.44	71.22 ± 15.44	73.26 ± 15.19
	NegGrad	44.78 ± 9.53	48.91 ± 9.60	49.47 ± 10.10	43.80 ± 11.20	38.23 ± 11.02	34.23 ± 8.57
	RandL	46.29 ± 11.93	47.89 ± 15.37	50.41 ± 17.50	52.53 ± 17.65	54.07 ± 15.93	53.48 ± 14.17
	Bad Teacher	25.22 ± 4.70	25.55 ± 5.01	24.58 ± 3.91	23.25 ± 5.12	19.22 ± 5.42	16.43 ± 6.06
	Amnesiac	23.86 ± 8.20	23.34 ± 7.86	23.03 ± 7.56	22.90 ± 8.15	22.51 ± 7.88	19.23 ± 8.16
	δ-Targeted[1]	**61.68 ± 15.20**	**61.02 ± 16.13**	59.60 ± 17.15	59.19 ± 17.17	59.62 ± 16.62	57.34 ± 17.32
	δ-Targeted[3]	60.84 ± 12.44	59.67 ± 13.10	**60.13 ± 14.65**	**60.05 ± 15.72**	**60.04 ± 16.49**	**59.98 ± 16.08**
$\boldsymbol{D_t}$	*BEFORE*	52.51 ± 0.23	52.51 ± 0.23	52.51 ± 0.23	52.51 ± 0.23	52.51 ± 0.23	52.53 ± 0.24
	NegGrad	39.32 ± 2.95	40.42 ± 2.47	40.13 ± 2.42	38.30 ± 2.62	32.08 ± 3.72	30.09 ± 2.27
	RandL	40.37 ± 3.05	40.58 ± 4.67	40.88 ± 5.49	42.48 ± 4.41	43.67 ± 3.74	44.17 ± 3.28
	Bad Teacher	24.66 ± 5.75	25.35 ± 5.11	26.43 ± 6.38	22.85 ± 5.12	18.95 ± 5.66	16.92 ± 6.53
	Amnesiac	22.98 ± 7.84	24.58 ± 9.00	23.35 ± 8.36	24.21 ± 9.51	23.13 ± 9.24	17.96 ± 8.23
	δ-Targeted[1]	**51.28 ± 1.30**	**50.68 ± 2.18**	49.79 ± 4.38	48.59 ± 2.98	47.97 ± 3.28	46.90 ± 3.68
	δ-Targeted[3]	50.04 ± 2.00	50.39 ± 1.17	**49.84 ± 3.36**	**50.77 ± 2.69**	**49.37 ± 2.29**	**47.02 ± 3.74**

in these tables are obtained by averaging over different neural network architectures, including ResNet-18, ResNet-50, and ViT.

The results for individual neural network architectures can be found in Figures 6.5 - 6.16.

Table 6.5: Means and standard deviations (percentage) of classification accuracy on CIFAR10 with ResNet-18.

	Method	$\|\boldsymbol{D}_f\|$=16	$\|\boldsymbol{D}_f\|$=32	$\|\boldsymbol{D}_f\|$=64	$\|\boldsymbol{D}_f\|$=128	$\|\boldsymbol{D}_f\|$=256	$\|\boldsymbol{D}_f\|$=512
$\boldsymbol{D_f}$	*BEFORE*	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	NegGrad	40.00 ± 48.99	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	7.50 ± 7.29	10.62 ± 4.24	11.56 ± 2.90	9.53 ± 2.39	10.23 ± 1.72	11.48 ± 1.18
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted^1	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted^3	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	99.99 ± 0.00	99.99 ± 0.00	99.99 ± 0.00	99.99 ± 0.00	99.99 ± 0.00	99.99 ± 0.00
	NegGrad	92.52 ± 7.43	74.05 ± 10.62	46.34 ± 7.70	25.25 ± 3.48	9.28 ± 1.58	4.97 ± 0.84
	RandL	92.17 ± 3.04	90.96 ± 1.34	83.47 ± 4.08	67.99 ± 3.01	50.06 ± 2.85	34.35 ± 1.25
	Bad Teacher	77.73 ± 4.66	81.09 ± 1.33	81.18 ± 1.27	77.40 ± 1.62	67.22 ± 4.14	45.04 ± 6.04
	Amnesiac	78.64 ± 1.34	79.30 ± 1.26	78.10 ± 1.46	74.23 ± 1.23	60.68 ± 1.34	35.88 ± 2.96
	δ-Targeted^1	**98.39 ± 1.05**	**98.83 ± 0.59**	96.61 ± 1.88	92.33 ± 1.88	79.59 ± 2.51	56.98 ± 2.33
	δ-Targeted^3	96.19 ± 1.65	97.91 ± 0.53	**97.57 ± 0.55**	**96.63 ± 0.62**	**91.43 ± 0.93**	**78.17 ± 0.89**
$\boldsymbol{D_t}$	*BEFORE*	93.13 ± 0.00	93.13 ± 0.00	93.13 ± 0.00	93.13 ± 0.00	93.13 ± 0.00	93.13 ± 0.00
	NegGrad	84.82 ± 7.64	66.76 ± 9.92	40.66 ± 6.60	21.56 ± 3.65	8.08 ± 1.33	4.72 ± 0.84
	RandL	84.33 ± 2.72	82.71 ± 1.18	75.24 ± 4.03	59.91 ± 2.60	43.35 ± 2.96	28.88 ± 1.21
	Bad Teacher	74.16 ± 3.86	76.79 ± 1.39	77.12 ± 0.87	74.00 ± 1.40	64.28 ± 4.14	43.15 ± 5.49
	Amnesiac	74.77 ± 1.19	75.50 ± 1.03	74.40 ± 1.48	70.59 ± 1.00	57.57 ± 1.35	34.47 ± 2.83
	δ-Targeted^1	**90.17 ± 1.21**	**90.72 ± 0.75**	88.19 ± 2.00	83.61 ± 2.05	71.06 ± 2.31	50.03 ± 1.96
	δ-Targeted^3	88.06 ± 1.55	89.64 ± 0.74	**89.30 ± 0.63**	**88.17 ± 0.62**	**82.68 ± 1.00**	**69.63 ± 1.05**

We first examine the performance on $\boldsymbol{D}_f$, which is the subset of data we want our model to unlearn. The tables show that all methods can effectively unlearn the data, resulting in near-zero accuracy on $\boldsymbol{D}_f$. Secondly, looking at the performance on $\boldsymbol{D}_r$, which is the remaining examples that the model should still remember, it is evident that both δ-Targeted^1 and δ-Targeted^3 consistently outperform the other methods. In particular, δ-Targeted^3 demonstrates superior performance compared to other methods, as it mostly achieves the highest accuracy across different sizes of $\boldsymbol{D}_f$, indicating that it effectively retains the model's ability to predict the remaining sample despite the unlearning process.

Lastly, on $\boldsymbol{D}_t$, the test dataset, both δ-Targeted^1 and δ-Targeted^3 again demonstrate superior performance. Despite the unlearning process, these methods ensure the model

Table 6.6: Means and standard deviations (percentage) of classification accuracy on CIFAR10 with ResNet-50.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	13.75 ± 10.00	10.62 ± 4.24	10.62 ± 3.19	11.72 ± 2.92	10.70 ± 1.09	10.78 ± 1.71
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	NegGrad	76.01 ± 4.78	64.18 ± 3.28	38.01 ± 6.06	17.06 ± 4.24	8.59 ± 1.43	4.94 ± 0.98
	RandL	92.51 ± 1.67	89.36 ± 2.70	79.43 ± 3.25	64.72 ± 4.14	47.95 ± 4.95	33.31 ± 2.80
	Bad Teacher	70.56 ± 2.01	68.54 ± 3.39	65.62 ± 1.04	54.88 ± 4.27	36.45 ± 3.05	23.18 ± 4.47
	Amnesiac	70.81 ± 0.30	70.05 ± 1.37	69.05 ± 1.63	64.36 ± 2.30	43.60 ± 1.78	26.51 ± 0.88
	δ-Targeted[1]	**98.92 ± 0.44**	**98.55 ± 0.51**	96.54 ± 0.73	90.86 ± 1.42	80.69 ± 3.27	58.80 ± 2.02
	δ-Targeted[3]	96.31 ± 0.74	96.96 ± 0.95	**97.23 ± 0.94**	**96.36 ± 0.91**	**92.09 ± 0.78**	**75.74 ± 0.82**
$\boldsymbol{D_t}$	*BEFORE*	94.02 ± 0.00	94.02 ± 0.00	94.02 ± 0.00	94.02 ± 0.00	94.02 ± 0.00	94.02 ± 0.00
	NegGrad	70.31 ± 4.60	59.22 ± 3.17	34.67 ± 5.74	15.26 ± 3.84	8.03 ± 1.40	4.53 ± 0.81
	RandL	85.60 ± 1.69	82.03 ± 2.49	72.26 ± 3.25	58.06 ± 3.52	42.72 ± 4.09	29.52 ± 2.69
	Bad Teacher	68.93 ± 2.02	66.55 ± 3.42	64.12 ± 1.23	53.19 ± 4.25	35.59 ± 3.11	22.50 ± 4.55
	Amnesiac	69.17 ± 0.40	68.22 ± 1.61	66.92 ± 1.61	62.33 ± 1.88	42.08 ± 2.14	25.42 ± 1.56
	δ-Targeted[1]	**91.55 ± 0.39**	**91.15 ± 0.42**	88.92 ± 0.74	83.04 ± 1.44	72.93 ± 3.24	52.51 ± 2.22
	δ-Targeted[3]	89.04 ± 0.61	89.75 ± 0.78	**89.90 ± 0.91**	**88.90 ± 0.98**	**84.37 ± 1.01**	**68.58 ± 0.88**

Table 6.7: Means and standard deviations (percentage) of classification accuracy on CIFAR10 with ViT.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	93.75 ± 6.85	94.38 ± 3.64	95.00 ± 3.62	95.16 ± 2.59	94.77 ± 2.27	93.48 ± 1.33
	NegGrad	8.75 ± 10.90	10.00 ± 2.34	8.75 ± 2.12	9.38 ± 2.21	10.86 ± 1.43	10.43 ± 1.41
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	12.50 ± 5.59	6.88 ± 4.59	12.19 ± 4.57	15.78 ± 5.49	18.59 ± 2.86	25.94 ± 2.20
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.55 ± 0.40	2.70 ± 1.01
$\boldsymbol{D_r}$	*BEFORE*	93.64 ± 0.00	93.64 ± 0.01	93.64 ± 0.01	93.64 ± 0.01	93.63 ± 0.01	93.64 ± 0.01
	NegGrad	40.34 ± 8.32	21.06 ± 6.47	14.10 ± 7.21	10.05 ± 0.10	9.99 ± 0.01	9.99 ± 0.01
	RandL	79.26 ± 1.48	74.43 ± 1.08	61.48 ± 3.34	47.20 ± 4.53	29.39 ± 1.79	18.96 ± 2.11
	Bad Teacher	85.98 ± 0.44	84.69 ± 0.55	80.46 ± 1.07	75.41 ± 0.97	65.98 ± 1.38	53.28 ± 3.38
	Amnesiac	81.04 ± 0.32	78.16 ± 0.53	74.15 ± 1.13	68.07 ± 1.19	58.34 ± 0.83	40.50 ± 2.02
	δ-Targeted[1]	85.74 ± 3.65	83.81 ± 2.10	74.86 ± 3.57	70.53 ± 3.32	65.69 ± 2.58	55.62 ± 2.66
	δ-Targeted[3]	**87.21 ± 1.78**	**85.96 ± 1.10**	**81.13 ± 2.18**	**78.66 ± 1.93**	**75.13 ± 2.29**	**67.42 ± 1.52**
$\boldsymbol{D_t}$	*BEFORE*	81.69 ± 0.00	81.69 ± 0.00	81.69 ± 0.00	81.69 ± 0.00	81.69 ± 0.00	81.69 ± 0.00
	NegGrad	38.11 ± 7.29	20.33 ± 5.93	13.80 ± 6.51	10.03 ± 0.11	10.00 ± 0.12	9.91 ± 0.16
	RandL	70.53 ± 1.21	65.85 ± 0.86	54.26 ± 3.02	42.25 ± 3.84	26.26 ± 1.69	16.90 ± 1.63
	Bad Teacher	**77.51 ± 0.52**	**76.52 ± 0.45**	**72.53 ± 0.82**	68.53 ± 0.59	60.51 ± 1.21	50.43 ± 2.82
	Amnesiac	72.48 ± 0.30	69.71 ± 0.48	66.16 ± 1.09	60.93 ± 0.87	52.44 ± 0.83	36.84 ± 1.96
	δ-Targeted[1]	75.47 ± 2.67	73.91 ± 1.47	66.41 ± 2.81	62.63 ± 2.84	58.08 ± 2.25	49.50 ± 2.40
	δ-Targeted[3]	76.68 ± 1.25	75.84 ± 0.81	71.66 ± 1.76	**69.43 ± 1.44**	**66.20 ± 1.97**	**59.58 ± 1.25**

Table 6.8: Means and standard deviations (percentage) of classification accuracy on CIFAR100 with ResNet-18.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.84 ± 0.31	99.92 ± 0.16	99.96 ± 0.08
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	20.00 ± 40.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	0.00 ± 0.00	0.62 ± 1.25	0.94 ± 1.25	0.47 ± 0.62	1.33 ± 0.19	0.86 ± 0.23
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	99.96 ± 0.00	99.96 ± 0.00	99.96 ± 0.00	99.96 ± 0.00	99.96 ± 0.00	99.96 ± 0.00
	NegGrad	83.71 ± 3.40	86.98 ± 0.84	84.97 ± 1.28	83.83 ± 8.28	70.57 ± 1.72	55.92 ± 0.95
	RandL	91.86 ± 2.90	95.32 ± 0.86	95.40 ± 0.57	94.58 ± 1.13	92.86 ± 0.40	89.79 ± 0.54
	Bad Teacher	36.13 ± 1.50	36.75 ± 1.51	35.45 ± 1.66	29.34 ± 2.15	25.57 ± 2.45	16.33 ± 2.24
	Amnesiac	35.47 ± 1.62	34.69 ± 0.65	33.22 ± 2.57	31.17 ± 0.89	22.39 ± 0.88	11.80 ± 0.63
	δ-Targeted[1]	**98.17 ± 0.90**	**98.99 ± 0.35**	**98.84 ± 0.10**	**98.16 ± 0.71**	**97.34 ± 0.37**	94.53 ± 0.56
	δ-Targeted[3]	94.89 ± 2.10	97.10 ± 0.69	97.05 ± 0.46	96.49 ± 1.05	96.06 ± 0.20	**95.36 ± 0.41**
$\boldsymbol{D_t}$	*BEFORE*	71.94 ± 0.00	71.94 ± 0.00	71.94 ± 0.00	71.94 ± 0.00	71.94 ± 0.00	71.94 ± 0.00
	NegGrad	56.30 ± 2.06	57.75 ± 0.63	56.30 ± 1.00	56.35 ± 7.57	46.14 ± 1.10	36.38 ± 0.44
	RandL	60.95 ± 2.05	64.02 ± 0.57	64.28 ± 0.36	63.21 ± 0.71	60.74 ± 0.46	56.39 ± 0.23
	Bad Teacher	30.66 ± 0.90	31.33 ± 1.25	30.04 ± 1.35	24.86 ± 1.82	21.32 ± 1.97	13.97 ± 1.77
	Amnesiac	30.54 ± 1.54	29.85 ± 0.98	28.35 ± 2.16	26.93 ± 0.94	19.26 ± 0.96	10.08 ± 0.42
	δ-Targeted[1]	**66.27 ± 1.38**	**68.06 ± 0.52**	**68.08 ± 0.16**	**67.36 ± 0.38**	**66.08 ± 0.30**	62.87 ± 0.71
	δ-Targeted[3]	63.38 ± 1.72	65.89 ± 0.60	66.52 ± 0.35	66.22 ± 0.60	65.45 ± 0.23	**64.06 ± 0.55**

Table 6.9: Means and standard deviations (percentage) of classification accuracy on CIFAR100 with ResNet-50.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.84 ± 0.31	99.92 ± 0.16	99.96 ± 0.08
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	0.00 ± 0.00	0.62 ± 1.25	1.25 ± 1.17	1.25 ± 0.80	1.33 ± 0.19	1.41 ± 0.19
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	99.97 ± 0.00	99.97 ± 0.00	99.97 ± 0.00	99.97 ± 0.00	99.97 ± 0.00	99.97 ± 0.00
	NegGrad	66.36 ± 7.53	66.55 ± 5.26	60.86 ± 3.08	57.38 ± 3.42	51.95 ± 3.63	43.16 ± 1.36
	RandL	87.54 ± 6.39	91.20 ± 2.52	92.83 ± 1.95	93.29 ± 1.47	92.44 ± 0.53	89.08 ± 0.85
	Bad Teacher	23.47 ± 1.07	23.83 ± 2.11	22.25 ± 0.71	11.25 ± 1.07	5.22 ± 0.82	3.36 ± 0.51
	Amnesiac	24.92 ± 0.98	25.12 ± 1.31	24.37 ± 0.62	21.51 ± 1.67	12.26 ± 1.60	7.36 ± 0.56
	δ-Targeted[1]	**98.36 ± 1.39**	**97.07 ± 1.26**	**97.27 ± 1.06**	**97.36 ± 0.66**	**96.87 ± 0.63**	93.58 ± 0.74
	δ-Targeted[3]	93.71 ± 3.49	94.68 ± 1.86	95.37 ± 1.43	95.55 ± 1.36	95.46 ± 0.53	**94.27 ± 0.67**
$\boldsymbol{D_t}$	*BEFORE*	74.67 ± 0.00	74.67 ± 0.00	74.67 ± 0.00	74.67 ± 0.00	74.67 ± 0.00	74.67 ± 0.00
	NegGrad	45.94 ± 4.63	45.44 ± 3.18	41.33 ± 2.02	38.76 ± 2.07	35.34 ± 2.76	29.79 ± 1.02
	RandL	60.14 ± 4.44	62.62 ± 2.06	64.22 ± 1.76	64.39 ± 1.40	62.79 ± 0.48	58.55 ± 0.41
	Bad Teacher	21.45 ± 0.90	21.28 ± 2.17	20.22 ± 0.75	9.70 ± 1.08	4.41 ± 0.74	2.62 ± 0.56
	Amnesiac	22.30 ± 0.69	22.33 ± 1.07	21.98 ± 0.69	19.08 ± 1.49	10.91 ± 1.58	6.56 ± 0.77
	δ-Targeted[1]	**69.77 ± 2.57**	**67.91 ± 1.18**	**68.89 ± 1.01**	**69.18 ± 0.87**	**68.55 ± 0.67**	64.65 ± 0.49
	δ-Targeted[3]	64.80 ± 3.68	65.76 ± 1.73	67.09 ± 1.40	67.48 ± 1.47	67.26 ± 0.43	**65.51 ± 0.60**

Table 6.10: Means and standard deviations (percentage) of classification accuracy on CIFAR100 with ViT.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	86.25 ± 8.29	81.88 ± 8.93	83.44 ± 2.54	84.84 ± 2.19	84.30 ± 0.62	84.84 ± 1.35
	NegGrad	1.25 ± 2.50	0.62 ± 1.25	0.62 ± 0.77	1.09 ± 0.38	0.78 ± 0.49	0.70 ± 0.47
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	3.75 ± 5.00	2.50 ± 2.34	4.06 ± 1.88	5.00 ± 1.45	6.72 ± 0.76	10.35 ± 0.54
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	85.18 ± 0.01	85.18 ± 0.01	85.19 ± 0.02	85.18 ± 0.00	85.19 ± 0.00	85.19 ± 0.01
	NegGrad	38.64 ± 2.76	25.91 ± 1.68	13.58 ± 3.10	1.55 ± 1.09	1.00 ± 0.00	1.00 ± 0.01
	RandL	66.80 ± 3.46	63.40 ± 2.19	56.68 ± 1.36	47.10 ± 2.25	36.13 ± 2.18	22.14 ± 0.82
	Bad Teacher	74.08 ± 0.38	71.23 ± 0.60	63.16 ± 1.72	52.82 ± 0.80	39.30 ± 1.25	26.99 ± 1.19
	Amnesiac	64.19 ± 1.46	62.44 ± 1.16	57.94 ± 1.15	53.96 ± 0.99	46.12 ± 0.75	31.17 ± 1.18
	δ-Targeted[1]	83.65 ± 0.52	82.49 ± 1.10	79.90 ± 0.99	76.41 ± 1.00	73.74 ± 1.49	70.48 ± 1.16
	δ-Targeted[3]	**84.04 ± 0.74**	**82.97 ± 1.15**	**80.86 ± 0.71**	**78.79 ± 0.71**	**77.57 ± 1.36**	**75.54 ± 0.65**
$\boldsymbol{D_t}$	*BEFORE*	54.59 ± 0.00	54.59 ± 0.00	54.59 ± 0.00	54.59 ± 0.00	54.59 ± 0.00	54.59 ± 0.00
	NegGrad	28.75 ± 1.70	20.25 ± 1.36	11.22 ± 2.61	1.45 ± 0.86	1.01 ± 0.06	0.98 ± 0.00
	RandL	45.19 ± 1.59	42.82 ± 1.23	38.65 ± 1.03	32.59 ± 1.49	25.17 ± 1.46	15.58 ± 0.43
	Bad Teacher	50.66 ± 0.48	49.06 ± 0.30	44.50 ± 0.66	38.84 ± 0.74	30.35 ± 1.05	22.14 ± 0.85
	Amnesiac	43.38 ± 0.59	42.34 ± 0.61	39.70 ± 0.59	37.25 ± 0.78	31.76 ± 0.50	22.45 ± 0.57
	δ-Targeted[1]	53.69 ± 0.20	53.25 ± 0.49	51.63 ± 0.59	49.86 ± 0.41	47.92 ± 0.94	46.00 ± 0.78
	δ-Targeted[3]	**53.92 ± 0.24**	**53.37 ± 0.36**	**52.14 ± 0.48**	**51.31 ± 0.32**	**50.21 ± 0.70**	**48.73 ± 0.70**

Table 6.11: Means and standard deviations (percentage) of classification accuracy on Fashion-MNIST with ResNet-18.

	Method	$\lvert\boldsymbol{D}_f\rvert$=16	$\lvert\boldsymbol{D}_f\rvert$=32	$\lvert\boldsymbol{D}_f\rvert$=64	$\lvert\boldsymbol{D}_f\rvert$=128	$\lvert\boldsymbol{D}_f\rvert$=256	$\lvert\boldsymbol{D}_f\rvert$=512
$\boldsymbol{D_f}$	*BEFORE*	95.00 ± 4.68	94.38 ± 3.64	94.69 ± 1.59	95.47 ± 0.91	96.02 ± 0.67	96.48 ± 0.86
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	12.50 ± 6.85	6.88 ± 2.34	8.44 ± 4.38	9.84 ± 3.51	12.89 ± 3.62	13.95 ± 5.09
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 1.37	3.24 ± 1.88
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	96.90 ± 0.00	96.90 ± 0.00	96.90 ± 0.01	96.90 ± 0.00	96.90 ± 0.00	96.90 ± 0.01
	NegGrad	73.65 ± 1.36	61.92 ± 3.62	45.79 ± 3.52	26.79 ± 0.93	16.05 ± 1.11	8.77 ± 1.49
	RandL	86.51 ± 1.35	85.38 ± 1.46	75.68 ± 1.65	60.56 ± 1.10	43.32 ± 1.95	28.76 ± 1.22
	Bad Teacher	86.49 ± 0.40	86.07 ± 0.29	85.71 ± 0.52	80.54 ± 2.86	68.33 ± 2.65	55.75 ± 3.67
	Amnesiac	85.68 ± 0.40	84.32 ± 0.77	81.41 ± 1.03	76.38 ± 1.07	59.85 ± 3.75	42.45 ± 3.57
	δ-Targeted[1]	**93.94 ± 0.50**	92.13 ± 0.68	87.06 ± 0.27	76.28 ± 1.39	59.01 ± 3.18	39.59 ± 1.16
	δ-Targeted[3]	92.93 ± 0.98	**93.35 ± 0.41**	**92.40 ± 0.57**	**87.61 ± 1.21**	**76.71 ± 1.94**	**58.28 ± 2.43**
$\boldsymbol{D_t}$	*BEFORE*	91.65 ± 0.00	91.65 ± 0.00	91.65 ± 0.00	91.65 ± 0.00	91.65 ± 0.00	91.65 ± 0.00
	NegGrad	69.50 ± 1.73	58.26 ± 4.21	42.21 ± 3.13	23.90 ± 0.98	13.82 ± 1.02	7.57 ± 1.16
	RandL	82.03 ± 1.07	81.17 ± 1.37	71.07 ± 2.18	56.25 ± 1.06	39.02 ± 2.11	24.99 ± 1.30
	Bad Teacher	85.18 ± 0.34	84.56 ± 0.36	84.27 ± 0.40	78.98 ± 2.81	67.08 ± 2.44	**55.04 ± 3.51**
	Amnesiac	84.37 ± 0.56	83.07 ± 0.54	80.05 ± 1.17	74.67 ± 0.90	58.42 ± 3.83	41.74 ± 3.85
	δ-Targeted[1]	**88.91 ± 0.48**	87.35 ± 0.70	82.36 ± 0.48	71.60 ± 2.03	54.48 ± 3.36	35.16 ± 1.24
	δ-Targeted[3]	88.12 ± 0.79	**88.57 ± 0.37**	**87.42 ± 0.58**	**82.74 ± 1.09**	**72.01 ± 1.92**	53.66 ± 2.49

Table 6.12: Means and standard deviations (percentage) of classification accuracy on Fashion-MNIST with ResNet-50.

	Method	$\|\boldsymbol{D}_f\|$=16	$\|\boldsymbol{D}_f\|$=32	$\|\boldsymbol{D}_f\|$=64	$\|\boldsymbol{D}_f\|$=128	$\|\boldsymbol{D}_f\|$=256	$\|\boldsymbol{D}_f\|$=512
$\boldsymbol{D}_f$	*BEFORE*	100.00 ± 0.00	98.75 ± 1.53	97.81 ± 1.59	97.66 ± 1.48	96.95 ± 1.06	97.30 ± 0.52
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	11.25 ± 7.29	11.25 ± 4.24	11.88 ± 2.90	10.94 ± 2.47	13.59 ± 1.79	12.66 ± 2.05
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 1.88	0.47 ± 0.94	6.25 ± 4.98	3.05 ± 1.36
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D}_r$	*BEFORE*	97.10 ± 0.00	97.10 ± 0.00	97.10 ± 0.00	97.10 ± 0.00	97.10 ± 0.00	97.10 ± 0.00
	NegGrad	68.70 ± 5.09	51.92 ± 6.19	34.08 ± 3.29	18.40 ± 1.46	9.01 ± 0.71	5.29 ± 0.41
	RandL	81.25 ± 6.89	75.48 ± 3.47	64.52 ± 4.04	46.83 ± 2.00	29.98 ± 1.67	20.67 ± 0.90
	Bad Teacher	83.84 ± 0.50	83.03 ± 1.65	77.70 ± 2.68	69.85 ± 4.52	51.28 ± 5.79	36.32 ± 3.92
	Amnesiac	83.74 ± 1.30	82.68 ± 0.98	78.56 ± 1.80	73.40 ± 0.83	60.89 ± 2.75	39.38 ± 1.59
	δ-Targeted[1]	**94.11 ± 0.63**	88.65 ± 2.09	80.80 ± 2.20	68.19 ± 3.23	51.23 ± 3.56	34.98 ± 1.17
	δ-Targeted[3]	93.96 ± 1.11	**93.56 ± 0.34**	**90.87 ± 1.12**	**83.50 ± 2.48**	**69.57 ± 1.73**	**49.50 ± 0.85**
$\boldsymbol{D}_t$	*BEFORE*	91.79 ± 0.00	91.79 ± 0.00	91.79 ± 0.00	91.79 ± 0.00	91.79 ± 0.00	91.79 ± 0.00
	NegGrad	64.50 ± 5.13	47.82 ± 6.14	30.65 ± 3.27	15.94 ± 1.63	7.76 ± 0.61	4.53 ± 0.26
	RandL	76.48 ± 6.88	70.32 ± 3.42	59.76 ± 4.00	42.36 ± 1.81	26.24 ± 1.62	17.54 ± 1.10
	Bad Teacher	82.79 ± 0.52	81.58 ± 1.94	76.59 ± 2.54	69.00 ± 4.55	50.54 ± 5.64	35.94 ± 3.93
	Amnesiac	82.81 ± 1.29	81.62 ± 1.12	77.85 ± 1.79	72.95 ± 0.79	60.62 ± 2.94	38.86 ± 1.98
	δ-Targeted[1]	89.19 ± 0.53	83.47 ± 2.55	75.66 ± 2.41	62.94 ± 3.07	46.41 ± 3.53	31.12 ± 0.94
	δ-Targeted[3]	**89.44 ± 0.73**	**88.87 ± 0.35**	**86.05 ± 1.28**	**78.45 ± 2.57**	**64.35 ± 1.61**	**45.15 ± 0.77**

Table 6.13: Means and standard deviations (percentage) of classification accuracy on Fashion-MNIST with ViT.

	Method	$\|\boldsymbol{D}_f\|$=16	$\|\boldsymbol{D}_f\|$=32	$\|\boldsymbol{D}_f\|$=64	$\|\boldsymbol{D}_f\|$=128	$\|\boldsymbol{D}_f\|$=256	$\|\boldsymbol{D}_f\|$=512
$\boldsymbol{D}_f$	*BEFORE*	93.75 ± 5.59	93.12 ± 4.59	91.88 ± 2.69	91.72 ± 1.17	92.42 ± 0.58	93.48 ± 0.64
	NegGrad	8.75 ± 9.35	8.12 ± 9.19	10.94 ± 6.63	6.72 ± 5.13	10.08 ± 2.87	10.43 ± 1.51
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	1.37 ± 0.35
	Bad Teacher	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Amnesiac	0.00 ± 0.00	1.25 ± 2.50	6.56 ± 2.50	18.44 ± 3.37	29.92 ± 2.82	25.74 ± 2.58
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	1.88 ± 1.17	3.28 ± 1.94	13.67 ± 3.91	52.50 ± 4.18
	δ-Targeted[3]	33.75 ± 8.48	31.25 ± 6.56	46.88 ± 6.01	67.19 ± 0.49	76.64 ± 2.35	77.54 ± 2.08
$\boldsymbol{D}_r$	*BEFORE*	94.11 ± 0.00	94.11 ± 0.00	94.11 ± 0.01	94.11 ± 0.00	94.11 ± 0.00	94.11 ± 0.01
	NegGrad	19.60 ± 7.44	9.94 ± 1.37	11.50 ± 4.18	8.39 ± 3.88	10.00 ± 0.01	10.00 ± 0.01
	RandL	48.39 ± 6.41	31.38 ± 5.46	20.56 ± 4.46	13.27 ± 2.49	8.74 ± 2.35	7.19 ± 1.05
	Bad Teacher	67.16 ± 9.11	59.51 ± 4.55	50.04 ± 3.26	39.13 ± 2.33	27.45 ± 1.21	21.05 ± 0.66
	Amnesiac	75.10 ± 6.75	73.45 ± 3.09	64.50 ± 2.36	57.24 ± 1.96	48.30 ± 1.54	38.16 ± 1.86
	δ-Targeted[1]	87.21 ± 0.54	84.90 ± 1.33	80.34 ± 1.15	73.08 ± 1.23	62.07 ± 1.20	61.94 ± 2.66
	δ-Targeted[3]	**90.53 ± 0.31**	**89.02 ± 0.67**	**86.85 ± 0.67**	**84.00 ± 0.38**	**80.82 ± 1.02**	**77.31 ± 1.44**
$\boldsymbol{D}_t$	*BEFORE*	92.41 ± 0.00	92.41 ± 0.00	92.41 ± 0.00	92.41 ± 0.00	92.41 ± 0.00	92.41 ± 0.00
	NegGrad	19.58 ± 7.47	10.03 ± 1.46	11.58 ± 4.10	8.51 ± 3.93	10.15 ± 0.12	10.06 ± 0.19
	RandL	47.47 ± 6.61	30.98 ± 5.32	20.20 ± 4.20	13.16 ± 2.61	8.71 ± 2.08	6.87 ± 0.96
	Bad Teacher	65.85 ± 9.02	58.35 ± 4.44	48.81 ± 3.24	37.86 ± 2.46	26.34 ± 1.54	20.44 ± 0.78
	Amnesiac	73.54 ± 6.76	71.91 ± 3.05	62.96 ± 2.56	55.70 ± 2.01	46.49 ± 1.51	36.23 ± 1.94
	δ-Targeted[1]	85.87 ± 0.77	83.72 ± 1.38	79.11 ± 1.22	71.81 ± 1.36	61.22 ± 1.35	61.48 ± 2.44
	δ-Targeted[3]	**89.09 ± 0.31**	**87.58 ± 0.71**	**85.36 ± 0.81**	**82.90 ± 0.57**	**80.05 ± 1.07**	**76.80 ± 1.38**

Table 6.14: Means and standard deviations (percentage) of classification accuracy on Tiny-ImageNet with ResNet-18.

	Method	$\|\boldsymbol{D}_f\|$=16	$\|\boldsymbol{D}_f\|$=32	$\|\boldsymbol{D}_f\|$=64	$\|\boldsymbol{D}_f\|$=128	$\|\boldsymbol{D}_f\|$=256	$\|\boldsymbol{D}_f\|$=512
$\boldsymbol{D_f}$	*BEFORE*	85.00 ± 6.37	85.62 ± 3.75	84.69 ± 2.07	84.84 ± 1.06	84.06 ± 1.17	84.61 ± 0.67
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	1.25 ± 2.50	0.62 ± 1.25	0.00 ± 0.00	0.62 ± 0.58	0.31 ± 0.46	0.55 ± 0.62
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.04 ± 0.08
$\boldsymbol{D_r}$	*BEFORE*	85.01 ± 0.01	85.01 ± 0.00	85.01 ± 0.00	85.01 ± 0.00	85.02 ± 0.00	85.02 ± 0.00
	NegGrad	58.03 ± 1.68	61.53 ± 1.86	63.25 ± 0.68	58.79 ± 0.71	53.26 ± 0.56	45.46 ± 1.72
	RandL	62.68 ± 1.90	69.22 ± 2.68	74.88 ± 1.21	76.67 ± 0.44	76.29 ± 0.48	73.07 ± 0.48
	Bad Teacher	18.82 ± 0.57	19.46 ± 0.65	19.73 ± 1.07	18.01 ± 0.79	12.37 ± 1.02	8.49 ± 0.56
	Amnesiac	12.62 ± 0.64	12.67 ± 0.54	12.68 ± 0.78	12.18 ± 0.57	11.63 ± 0.54	8.04 ± 0.48
	δ-Targeted[1]	**83.02 ± 1.07**	**83.43 ± 0.52**	**83.60 ± 0.38**	**83.36 ± 0.26**	**82.93 ± 0.18**	81.70 ± 0.35
	δ-Targeted[3]	76.88 ± 4.04	77.96 ± 2.31	80.45 ± 0.71	81.89 ± 0.33	82.77 ± 0.22	**82.51 ± 0.20**
$\boldsymbol{D_t}$	*BEFORE*	52.71 ± 0.00	52.71 ± 0.00	52.71 ± 0.00	52.71 ± 0.00	52.71 ± 0.00	52.71 ± 0.00
	NegGrad	40.47 ± 1.05	42.26 ± 0.87	42.37 ± 0.81	39.40 ± 0.77	35.90 ± 0.42	31.02 ± 0.97
	RandL	43.58 ± 1.17	46.22 ± 1.31	48.25 ± 0.42	48.39 ± 0.40	47.61 ± 0.49	45.15 ± 0.38
	Bad Teacher	17.81 ± 0.62	18.90 ± 0.51	18.87 ± 1.42	17.08 ± 1.00	11.72 ± 0.97	7.88 ± 0.61
	Amnesiac	12.22 ± 0.46	12.15 ± 0.63	12.07 ± 0.72	11.40 ± 0.42	10.65 ± 0.82	7.10 ± 0.62
	δ-Targeted[1]	**52.00 ± 0.34**	**52.18 ± 0.23**	**52.04 ± 0.24**	**51.75 ± 0.30**	**51.57 ± 0.25**	50.97 ± 0.35
	δ-Targeted[3]	49.63 ± 1.32	50.08 ± 0.64	50.97 ± 0.49	51.24 ± 0.41	51.40 ± 0.40	**51.37 ± 0.20**

Table 6.15: Means and standard deviations (percentage) of classification accuracy on Tiny-ImageNet with ResNet-50.

	Method	$\|\boldsymbol{D}_f\|$=16	$\|\boldsymbol{D}_f\|$=32	$\|\boldsymbol{D}_f\|$=64	$\|\boldsymbol{D}_f\|$=128	$\|\boldsymbol{D}_f\|$=256	$\|\boldsymbol{D}_f\|$=512
$\boldsymbol{D_f}$	*BEFORE*	28.12 ± 9.38	29.69 ± 4.69	28.91 ± 0.78	30.86 ± 5.08	32.23 ± 3.71	33.59 ± 0.00
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.31 ± 0.38	0.31 ± 0.29	0.35 ± 0.19
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[1]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted[3]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	53.04 ± 0.02	53.05 ± 0.00	53.05 ± 0.00	53.06 ± 0.00	53.05 ± 0.00	53.06 ± 0.00
	NegGrad	38.70 ± 1.30	42.01 ± 0.85	40.98 ± 0.73	36.26 ± 1.20	30.46 ± 0.75	27.95 ± 0.96
	RandL	39.28 ± 0.61	38.70 ± 0.88	39.13 ± 0.67	40.85 ± 0.63	42.73 ± 0.38	43.96 ± 0.85
	Bad Teacher	28.46 ± 0.77	27.44 ± 2.26	28.75 ± 0.56	26.27 ± 0.32	21.78 ± 0.28	21.22 ± 0.29
	Amnesiac	29.35 ± 0.61	29.50 ± 0.46	29.59 ± 0.57	29.07 ± 0.39	27.57 ± 0.35	25.02 ± 0.42
	δ-Targeted[1]	51.35 ± 0.68	50.34 ± 0.60	49.27 ± 0.43	47.84 ± 1.11	46.69 ± 0.84	44.81 ± 0.54
	δ-Targeted[3]	**51.65 ± 0.81**	**51.29 ± 0.33**	**50.74 ± 0.20**	**49.88 ± 0.47**	**48.84 ± 0.61**	**46.95 ± 0.80**
$\boldsymbol{D_t}$	*BEFORE*	52.17 ± 0.00	52.17 ± 0.00	52.17 ± 0.00	52.17 ± 0.00	52.17 ± 0.00	52.17 ± 0.00
	NegGrad	37.86 ± 0.91	41.03 ± 0.91	40.44 ± 0.85	35.77 ± 1.19	29.85 ± 0.83	27.48 ± 0.81
	RandL	38.51 ± 0.72	38.15 ± 0.69	38.67 ± 0.63	40.35 ± 0.34	41.80 ± 0.37	42.68 ± 1.00
	Bad Teacher	28.60 ± 1.01	27.45 ± 2.33	28.70 ± 0.88	26.14 ± 0.50	21.80 ± 0.16	21.20 ± 0.37
	Amnesiac	29.36 ± 0.65	29.38 ± 0.43	29.65 ± 0.64	28.93 ± 0.43	27.40 ± 0.37	24.67 ± 0.53
	δ-Targeted[1]	50.54 ± 0.80	49.73 ± 0.68	48.46 ± 0.34	46.88 ± 1.11	45.77 ± 0.78	44.15 ± 0.62
	δ-Targeted[3]	**50.89 ± 0.90**	**50.68 ± 0.39**	**50.02 ± 0.19**	**48.97 ± 0.60**	**47.86 ± 0.63**	**46.25 ± 0.83**

Table 6.16: Means and standard deviations (percentage) of classification accuracy on Tiny-ImageNet with ViT.

	Method	$\|\boldsymbol{D}_f\|$=16	$\|\boldsymbol{D}_f\|$=32	$\|\boldsymbol{D}_f\|$=64	$\|\boldsymbol{D}_f\|$=128	$\|\boldsymbol{D}_f\|$=256	$\|\boldsymbol{D}_f\|$=512
$\boldsymbol{D_f}$	*BEFORE*	49.75 ± 1.25	49.50 ± 0.50	50.05 ± 2.25	50.10 ± 0.40	51.55 ± 1.45	47.00 ± 0.00
	NegGrad	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	RandL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Bad Teacher	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.31 ± 0.38	0.31 ± 0.29	0.35 ± 0.19
	Amnesiac	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted^1	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	δ-Targeted^3	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$\boldsymbol{D_r}$	*BEFORE*	54.89 ± 0.00	54.89 ± 0.00	54.89 ± 0.00	54.89 ± 0.00	54.89 ± 0.00	54.89 ± 0.00
	NegGrad	37.60 ± 1.96	43.18 ± 5.69	44.17 ± 3.90	36.36 ± 6.14	30.96 ± 4.92	29.29 ± 5.16
	RandL	36.90 ± 4.11	35.76 ± 3.72	37.21 ± 4.06	40.06 ± 7.68	43.20 ± 4.56	43.41 ± 5.04
	Bad Teacher	28.36 ± 2.01	29.74 ± 3.41	25.26 ± 1.75	25.49 ± 6.03	23.50 ± 3.93	19.59 ± 3.70
	Amnesiac	29.61 ± 3.36	27.86 ± 3.56	26.82 ± 2.44	27.45 ± 5.00	28.32 ± 2.86	24.63 ± 3.36
	δ-Targeted^1	50.67 ± 2.84	49.30 ± 5.14	45.94 ± 3.66	46.37 ± 2.44	**49.24 ± 3.04**	45.51 ± 2.92
	δ-Targeted^3	**54.00 ± 7.63**	**49.75 ± 2.40**	**49.18 ± 4.68**	**48.39 ± 4.99**	48.51 ± 6.30	**50.47 ± 2.64**
$\boldsymbol{D_t}$	*BEFORE*	52.34 ± 0.00	52.34 ± 0.00	52.34 ± 0.00	52.34 ± 0.00	52.34 ± 0.00	52.34 ± 0.00
	NegGrad	39.64 ± 4.54	37.96 ± 2.64	37.58 ± 2.14	39.75 ± 2.97	30.48 ± 4.31	31.76 ± 1.83
	RandL	39.02 ± 3.22	37.39 ± 3.91	35.73 ± 2.03	38.69 ± 2.03	41.61 ± 4.28	44.66 ± 5.27
	Bad Teacher	27.57 ± 5.18	29.71 ± 2.74	31.72 ± 5.41	25.34 ± 5.20	23.34 ± 3.93	21.68 ± 2.21
	Amnesiac	27.35 ± 2.90	32.21 ± 2.56	28.33 ± 4.09	32.29 ± 4.37	31.35 ± 3.69	22.13 ± 4.70
	δ-Targeted^1	**51.31 ± 1.81**	50.13 ± 3.21	**48.87 ± 7.05**	47.13 ± 3.20	46.59 ± 3.44	**45.59 ± 3.78**
	δ-Targeted^3	49.60 ± 2.88	**50.42 ± 1.82**	48.52 ± 5.52	**52.09 ± 3.99**	**48.85 ± 2.92**	43.44 ± 2.96

maintains its overall predictive performance on unseen data, outperforming other methods across different sizes of $\boldsymbol{D}_f$.

When the size of the unlearning sample $\boldsymbol{D}_f$ is large, δ-Targeted^3 consistently outperforms the other methods. It effectively unlearns harder-to-forget examples by emphasising the importance of harder-to-forget examples in the unlearning set. This makes it particularly advantageous for larger unlearning sets. However, when the unlearning set is small, δ-Targeted^1 tends to outperform δ-Targeted^3. This might be because when all examples can be easily unlearned, placing extra emphasis on certain examples (as in $\lambda = 3$) does not provide additional benefits. A potential reason could be that, when the to-be-forgotten sample size is small, these examples can be unlearned in just a few steps. Emphasising the loss too much at the beginning can lead to large parameter updates, which adversely affect the model's performance on the remaining data.

6.5.6 Sequential Unlearning for Sequential Recommender Systems

The proposed δ-Targeted$^{\gamma}$ label noise injection method is a general machine unlearning method. It can be combined with many real-world applications, such as recommender systems. As we have stated, δ-Targeted$^{\gamma}$ will greatly reduce the performance degeneration while implementing unlearning. We would like to combine it with the unlearning method proposed in Chapter 5 and see how it works.

The experiment setups and the evaluation metrics in this section are the same as those used in Chapter 5. The comparisons are shown in Figures 6.17 - 6.20. We can see that the newly proposed δ-Targeted$^{\gamma}$ consistently outperforms DyRand and other baselines on four sequential recommender systems, i.e., Beauty, Sports and Outdoors, Toys and Games, and Yelp datasets under different metrics.

The experimental results clearly show that we have successfully unlearn some requested data points while greatly maintaining the model's performance. Our approach achieves the state-of-the-art method to maintain the models' utility function while protecting users' privacy. This is an important direction that warrants further exploration to build trustworthy recommender systems.

Some results show that the unlearned model can sometimes outperform the original model on remaining data by a very small margin. This might be the regularisation effect of the proposed method, which prevents the model from overfitting.

Table 6.17: HIT@10, NDCG@10 and MRR by applying different unlearning methods to SASRec on Beauty dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	40.00	30.00	33.33	40.00	40.00	43.33	45.71	45.00
		NDCG@10	18.93	14.46	14.23	15.84	14.98	17.26	18.15	18.10
		MRR	16.66	14.33	12.81	12.37	11.36	13.06	13.31	13.34
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	56.28	56.29	56.30	56.30	56.30	56.30	56.30	56.30
		NDCG@10	27.24	27.24	27.24	27.24	27.25	27.25	27.25	27.25
		MRR	20.79	20.79	20.79	20.79	20.80	20.80	20.80	20.80
	NegGrad	HIT@10	45.03	43.00	32.72	43.67	19.66	41.12	35.66	32.20
		NDCG@10	22.39	21.30	16.25	21.38	10.02	20.35	17.86	16.15
		MRR	17.66	16.85	13.15	16.85	8.37	16.12	14.36	13.04
	FixRand	HIT@10	50.77	51.42	50.91	51.00	49.17	49.65	45.29	47.71
		NDCG@10	24.75	25.51	25.14	25.25	25.12	25.28	25.04	24.52
		MRR	19.20	19.91	19.61	19.55	19.54	19.74	19.21	19.38
	DyRand	HIT@10	51.16	51.94	51.45	51.14	50.91	50.69	50.10	47.96
		NDCG@10	25.23	25.72	25.45	25.42	25.04	25.32	24.99	24.77
		MRR	19.65	19.99	19.82	19.61	19.75	20.09	19.65	19.71
	δ-Targeted[1]	HIT@10	51.38	52.12	51.74	51.46	51.25	51.03	50.48	48.32
		NDCG@10	25.48	26.01	25.72	25.70	25.32	25.60	25.26	25.05
		MRR	19.92	20.28	20.06	19.88	20.01	20.33	19.96	20.02
	δ-Targeted[3]	HIT@10	**51.62**	**52.35**	**51.98**	**51.71**	**51.49**	**51.26**	**50.71**	**48.58**
		NDCG@10	**25.66**	**26.18**	**25.91**	**25.88**	**25.51**	**25.78**	**25.45**	**25.23**
		MRR	**20.08**	**20.41**	**20.19**	**20.02**	**20.17**	**20.48**	**20.12**	**20.20**
D_t (↑)	BEFORE	HIT@10	5.68	5.68	5.68	5.68	5.68	5.68	5.68	5.68
		NDCG@10	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.03
		MRR	2.50	2.50	2.50	2.50	2.50	2.50	2.50	2.50
	NegGrad	HIT@10	4.94	5.09	4.34	4.90	2.44	4.97	4.22	3.76
		NDCG@10	2.64	2.71	2.26	2.60	1.30	2.66	2.30	2.03
		MRR	2.20	2.23	1.85	2.16	1.11	2.21	1.94	1.68
	FixRand	HIT@10	5.54	5.50	5.37	5.42	5.25	5.23	4.99	5.04
		NDCG@10	2.94	2.94	2.90	2.91	2.86	2.80	2.65	2.71
		MRR	2.40	2.44	2.43	2.42	2.40	2.33	2.19	2.27
	DyRand	HIT@10	5.59	5.52	5.42	5.42	5.35	5.29	5.25	4.97
		NDCG@10	2.94	2.97	2.91	2.90	2.90	2.89	2.83	2.69
		MRR	2.40	2.47	2.43	2.41	2.42	2.43	2.36	2.25
	δ-Targeted[1]	HIT@10	**5.81**	5.68	5.77	5.60	5.63	5.59	5.63	5.60
		NDCG@10	**2.98**	2.92	2.94	2.90	2.91	2.89	2.90	2.87
		MRR	2.45	2.42	2.42	2.43	2.43	2.43	2.41	2.38
	δ-Targeted[3]	HIT@10	5.76	**5.78**	**5.79**	**5.75**	**5.75**	**5.76**	**5.76**	**5.79**
		NDCG@10	**2.98**	**2.98**	**2.99**	**2.97**	**2.97**	**2.98**	**2.98**	**2.98**
		MRR	**2.48**	**2.47**	**2.48**	**2.48**	**2.48**	**2.48**	**2.48**	**2.47**

Table 6.18: HIT@10, NDCG@10 and MRR by applying different unlearning methods to SASRec on Sports and Outdoors dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	0.00	0.00	6.67	5.00	4.00	3.33	2.86	5.00
		NDCG@10	0.00	0.00	4.21	3.15	2.52	2.10	1.80	2.30
		MRR	0.00	0.00	4.42	3.70	2.96	2.72	2.33	2.55
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	6.73	6.73	6.73	6.73	6.73	6.73	6.73	6.73
		NDCG@10	3.33	3.33	3.33	3.33	3.33	3.33	3.33	3.33
		MRR	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10
	NegGrad	HIT@10	6.43	6.21	4.79	4.06	4.26	2.96	3.26	2.74
		NDCG@10	3.28	3.24	2.35	2.36	2.06	1.51	1.67	1.42
		MRR	2.57	2.53	2.13	2.03	1.83	1.33	1.48	1.26
	FixRand	HIT@10	6.68	6.34	6.31	5.82	5.88	5.77	5.26	5.14
		NDCG@10	3.20	3.10	2.96	2.62	2.54	2.53	2.18	2.05
		MRR	3.02	2.93	2.82	2.47	2.39	2.22	2.08	1.92
	DyRand	HIT@10	**6.71**	6.60	6.52	6.42	**6.33**	5.86	5.45	5.17
		NDCG@10	3.22	3.20	3.15	2.97	2.70	2.61	2.41	2.30
		MRR	**3.10**	3.03	2.98	2.86	2.61	2.43	2.22	2.18
	δ-Targeted[1]	HIT@10	6.56	6.54	5.14	5.20	4.98	4.96	4.79	5.57
		NDCG@10	3.26	3.25	2.58	2.61	2.51	2.52	2.40	2.72
		MRR	3.03	3.02	2.35	2.40	2.26	2.29	2.16	2.47
	δ-Targeted[3]	HIT@10	6.68	**6.69**	**6.56**	**6.55**	6.27	**6.21**	**6.17**	**6.67**
		NDCG@10	**3.31**	**3.32**	**3.26**	**3.24**	**3.08**	**3.08**	**3.05**	**3.30**
		MRR	3.07	**3.09**	**3.04**	**3.01**	**2.85**	**2.84**	**2.83**	**3.06**
D_t (↑)	BEFORE	HIT@10	2.59	2.59	2.59	2.59	2.59	2.59	2.59	2.59
		NDCG@10	1.37	1.37	1.37	1.37	1.37	1.37	1.37	1.37
		MRR	1.19	1.19	1.19	1.19	1.19	1.19	1.19	1.19
	NegGrad	HIT@10	2.42	2.24	1.72	2.45	1.60	1.46	1.59	1.11
		NDCG@10	1.27	1.19	0.88	1.26	0.80	0.71	0.82	0.55
		MRR	1.11	1.05	0.78	1.08	0.69	0.60	0.71	0.47
	FixRand	HIT@10	2.57	2.57	2.38	2.32	2.26	2.16	2.03	1.84
		NDCG@10	1.37	1.37	1.26	1.22	1.19	1.14	1.06	0.97
		MRR	1.19	1.19	1.08	1.04	1.00	0.97	0.89	0.82
	DyRand	HIT@10	2.56	2.58	2.39	2.54	2.28	2.31	2.16	2.01
		NDCG@10	1.37	1.37	1.27	1.30	1.19	1.17	1.11	1.02
		MRR	1.20	1.20	1.10	1.10	1.02	0.97	0.92	0.85
	δ-Targeted[1]	HIT@10	2.58	2.56	2.30	2.28	2.29	2.22	2.22	2.24
		NDCG@10	1.37	1.37	1.17	1.14	1.18	1.14	1.13	1.17
		MRR	1.19	1.20	1.01	0.97	1.01	0.98	0.97	1.02
	δ-Targeted[3]	HIT@10	**2.59**	**2.60**	**2.57**	**2.58**	**2.52**	**2.53**	**2.47**	**2.59**
		NDCG@10	**1.39**	**1.38**	**1.36**	**1.37**	**1.33**	**1.34**	**1.31**	**1.37**
		MRR	**1.21**	**1.20**	**1.18**	**1.19**	**1.16**	**1.17**	**1.15**	**1.19**

Table 6.19: HIT@10, NDCG@10 and MRR by applying different unlearning methods to SASRec on Toys and Games dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	20.00	30.00	33.33	35.00	40.00	36.67	34.29	32.50
		NDCG@10	12.62	14.48	18.55	17.84	20.68	18.67	17.81	16.83
		MRR	10.87	11.05	14.99	13.78	16.33	14.92	14.51	13.62
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	36.24	36.24	36.24	36.24	36.23	36.23	36.24	36.24
		NDCG@10	17.46	17.46	17.46	17.46	17.46	17.46	17.46	17.46
		MRR	13.89	13.89	13.88	13.89	13.88	13.88	13.88	13.89
	NegGrad	HIT@10	34.29	29.77	24.46	2.79	5.83	13.37	19.14	3.45
		NDCG@10	16.47	14.48	12.13	1.34	2.76	6.49	9.41	1.62
		MRR	12.80	11.77	9.83	1.17	2.38	5.57	7.81	1.35
	DyRand	HIT@10	**35.49**	34.39	34.13	33.29	29.89	24.53	22.83	17.28
		NDCG@10	**16.90**	**16.77**	16.27	16.40	16.34	13.52	12.63	9.57
		MRR	**13.84**	**13.59**	**13.95**	13.03	13.30	11.11	10.42	7.95
	FixRand	HIT@10	34.89	34.37	34.22	34.02	34.13	33.39	30.03	28.58
		NDCG@10	16.62	16.63	16.77	16.68	16.68	16.50	16.00	14.98
		MRR	13.67	13.38	13.41	13.10	13.04	13.09	13.04	12.16
	δ-Targeted[1]	HIT@10	33.16	32.20	30.20	30.32	29.48	28.53	28.16	28.25
		NDCG@10	15.85	15.32	14.35	14.35	14.02	13.53	13.30	13.42
		MRR	12.67	12.28	11.65	11.62	11.44	11.09	10.88	10.98
	δ-Targeted[3]	HIT@10	34.92	**34.98**	**35.38**	**34.98**	**36.22**	**36.21**	**36.20**	**34.94**
		NDCG@10	16.74	16.76	**16.92**	**16.76**	**17.46**	**17.49**	**17.48**	**16.81**
		MRR	13.34	13.34	13.45	**13.34**	**13.86**	**13.90**	**13.90**	**13.42**
D_t (↑)	BEFORE	HIT@10	6.71	6.71	6.71	6.71	6.71	6.71	6.71	6.71
		NDCG@10	3.74	3.74	3.74	3.74	3.74	3.74	3.74	3.74
		MRR	3.12	3.12	3.12	3.12	3.12	3.12	3.12	3.12
	NegGrad	HIT@10	6.07	5.91	5.08	1.14	2.28	4.11	4.71	1.46
		NDCG@10	3.33	3.19	2.66	0.58	1.12	2.14	2.44	0.78
		MRR	2.77	2.64	2.20	0.52	0.95	1.80	2.00	0.70
	FixRand	HIT@10	**6.72**	**6.71**	6.14	5.38	5.34	4.84	4.62	3.89
		NDCG@10	3.72	**3.76**	3.44	2.97	2.99	2.66	2.56	2.11
		MRR	3.10	**3.14**	2.89	2.46	2.50	2.20	2.13	1.75
	DyRand	HIT@10	6.70	6.59	6.33	5.83	5.85	5.77	5.26	5.22
		NDCG@10	**3.73**	3.64	3.46	3.20	3.26	3.14	2.95	2.90
		MRR	**3.12**	3.02	2.86	2.66	2.73	2.58	2.49	2.43
	δ-Targeted[1]	HIT@10	6.56	6.44	6.18	6.23	6.08	6.02	6.01	6.05
		NDCG@10	3.59	3.55	3.35	3.41	3.36	3.29	3.27	3.28
		MRR	2.99	2.97	2.78	2.85	2.84	2.75	2.72	2.74
	δ-Targeted[3]	HIT@10	6.61	6.61	**6.63**	**6.63**	**6.65**	**6.65**	**6.65**	**6.61**
		NDCG@10	3.66	3.65	**3.66**	**3.66**	**3.72**	**3.71**	**3.71**	**3.66**
		MRR	3.06	3.04	**3.05**	**3.05**	**3.11**	**3.11**	**3.11**	**3.06**

Table 6.20: HIT@10, NDCG@10 and MRR by applying different unlearning methods to SASRec on Yelp dataset.

	Method	Metric	$\|D_f\|$=5	$\|D_f\|$=10	$\|D_f\|$=15	$\|D_f\|$=20	$\|D_f\|$=25	$\|D_f\|$=30	$\|D_f\|$=35	$\|D_f\|$=40
D_f (↓)	BEFORE	HIT@10	0.00	10.00	6.67	5.00	4.00	3.33	5.71	5.00
		NDCG@10	0.00	3.56	2.37	1.78	1.42	1.19	3.87	3.39
		MRR	0.00	1.92	1.28	0.96	0.77	0.72	3.56	3.12
	All Others	HIT@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		NDCG@10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MRR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
D_r (↑)	BEFORE	HIT@10	4.26	4.26	4.26	4.26	4.26	4.26	4.26	4.26
		NDCG@10	2.20	2.20	2.20	2.20	2.20	2.20	2.20	2.20
		MRR	1.96	1.96	1.96	1.96	1.96	1.96	1.96	1.96
	NegGrad	HIT@10	**4.50**	4.02	5.00	4.14	4.82	4.27	3.57	0.38
		NDCG@10	**2.29**	2.07	2.51	2.15	2.49	2.17	1.98	0.21
		MRR	**1.99**	1.86	2.17	1.91	2.21	1.92	1.73	0.20
	FixRand	HIT@10	4.31	5.42	5.03	**5.59**	5.29	5.86	3.05	3.23
		NDCG@10	2.22	**2.80**	2.57	**2.88**	2.70	3.06	1.60	1.65
		MRR	1.98	**2.49**	2.32	**2.52**	2.39	2.71	1.39	1.42
	DyRand	HIT@10	4.31	**5.43**	**5.76**	5.44	**5.65**	**6.11**	**4.24**	**4.20**
		NDCG@10	2.22	2.79	**2.98**	2.81	**2.93**	**3.20**	**2.18**	**2.15**
		MRR	1.98	**2.49**	**2.63**	2.50	**2.59**	**2.81**	1.87	1.85
	δ-Targeted[1]	HIT@10	4.25	4.28	3.09	2.91	4.19	4.18	4.17	4.10
		NDCG@10	2.20	2.19	1.60	1.52	2.17	2.16	2.16	2.12
		MRR	1.96	1.95	1.45	1.38	1.94	1.93	**1.92**	**1.90**
	δ-Targeted[3]	HIT@10	4.30	4.27	4.27	3.84	4.24	4.25	3.77	3.67
		NDCG@10	2.21	2.20	2.20	1.94	2.19	2.20	1.95	1.88
		MRR	1.96	1.96	1.96	1.73	1.96	1.96	1.76	1.68
D_t (↑)	BEFORE	HIT@10	2.75	2.75	2.75	2.75	2.75	2.75	2.75	2.75
		NDCG@10	1.40	1.40	1.40	1.40	1.40	1.40	1.40	1.40
		MRR	1.20	1.20	1.20	1.20	1.20	1.20	1.20	1.20
	NegGrad	HIT@10	2.26	2.31	2.15	1.98	2.10	2.30	1.25	0.16
		NDCG@10	1.14	1.16	1.09	1.00	1.06	1.14	0.62	0.08
		MRR	0.98	1.01	0.94	0.85	0.91	0.97	0.52	0.08
	FixRand	HIT@10	2.74	2.49	2.50	2.40	2.43	1.91	1.30	1.28
		NDCG@10	1.39	1.26	1.27	1.21	1.25	0.82	0.66	0.64
		MRR	1.19	1.07	1.08	1.03	1.07	0.69	0.57	0.55
	DyRand	HIT@10	2.75	2.52	2.60	2.44	2.47	2.50	1.42	1.29
		NDCG@10	1.39	1.27	1.32	1.26	1.27	1.27	0.71	0.64
		MRR	1.19	1.09	1.13	1.09	1.10	1.08	0.61	0.55
	δ-Targeted[1]	HIT@10	**2.78**	**2.81**	2.11	2.08	**2.77**	**2.79**	**2.79**	**2.79**
		NDCG@10	**1.41**	**1.43**	1.08	1.07	**1.42**	**1.42**	**1.41**	**1.42**
		MRR	**1.22**	**1.23**	0.95	0.94	**1.22**	**1.22**	**1.21**	**1.22**
	δ-Targeted[3]	HIT@10	2.77	2.80	**2.79**	**2.58**	**2.79**	**2.79**	2.68	2.60
		NDCG@10	1.40	1.42	**1.41**	**1.28**	**1.42**	1.41	1.37	1.31
		MRR	1.20	1.21	**1.21**	**1.09**	**1.22**	1.21	1.18	1.13

6.6 Summary

In this Chapter, we presented δ-Targeted$^{\gamma}$, a method designed to retain model performance after removing a specific training sample without the need for reassessing the remaining sample. Theoretically, we demonstrated that our method results in more concentrated parameter updates than those introduced by gradient ascent, thereby effectively reducing the performance drops commonly observed after unlearning. Our empirical evaluations further corroborate these findings, as δ-Targeted$^{\gamma}$ consistently achieves state-of-the-art results across a variety of benchmark image datasets, underscoring both its theoretical soundness and practical efficacy. It also helps recommender systems maintain high utility while preserving privacy.

CHAPTER 7

CONCLUSION AND FUTURE RESEARCH

7.1 Conclusion

In this thesis, we have delved into the challenges of building trustworthy recommender systems, achieving two primary goals: developing robust recommender system algorithms to handle low-quality input data and designing efficient mechanisms to protect user privacy through machine unlearning. By addressing the vulnerabilities inherent in modern recommendation pipelines, i.e., ranging from limited feedback and corrupted ratings to the "right to be forgotten", this research establishes a framework for recommender systems that are both reliable and ethically compliant.

To enhance the reliability of recommendations under data adversity, we identified and tackled two newly exposed challenges. First, we investigated the limited feedback problem, where explicit acquisition signals are unavailable. We proposed the Partial Acquisition Recommender System (PARS), which leverages masked language modelling and partial label learning to extract reliable user preferences from browsing histories alone. Second, we addressed rating flip noise by employing a label noise transition matrix

to model the probability of rating corruption. By applying a statistically risk-consistent algorithm for loss correction, we successfully trained robust models that maintain high accuracy even when the interaction history is significantly corrupted.

Furthermore, we extended our investigation to the paramount concern of user privacy and data sovereignty. We addressed the challenge of machine unlearning for recommender systems, particularly in settings where retraining from scratch or accessing the remaining training data is impractical. We developed two core strategies: (1) Sequence Unlearning via Label Noise Injection. It is a dynamic approach for sequential recommender systems that encourages random predictions for "to-be-forgotten" sequences, preventing model overfitting and parameter distortion. (2) δ-Targeted$^{\gamma}$ unlearning. It is a method that assigns controllable, incorrect labels to specific training samples. We theoretically and empirically demonstrated that this approach results in more concentrated parameter updates than traditional gradient ascent, effectively preserving model utility while ensuring data removal.

In conclusion, our research offers a comprehensive suite of methods to build recommender systems that remain accurate under data noise and respectful of user privacy. These advancements contribute to the evolution of intelligent systems that foster user trust and long-term engagement in dynamic, privacy-conscious environments.

7.2 Future Research

This thesis identifies the following directions as critical areas for future work:

- **Establishing Theoretical Guarantees**: While our methods demonstrate strong empirical performance, future work will focus on building more rigorous theoretical foundations. This includes investigating the generalisation bounds of PARS and our unlearning algorithms under diverse architectural constraints.

- **Modelling Context-Dependent Noise**: Our current rating flip noise model primarily depends on the rating set. Future research will explore more complex, context-dependent noise models where corruption probabilities vary based on specific user profiles, item characteristics, and or temporal contexts.

- **Hybrid Learning with Sparse Labels**: We aim to investigate semi-supervised extensions of the PARS framework. Specifically, we will study how to effectively integrate a small fraction of explicit acquisition labels (when available) with large-scale browsing data to further boost predictive power.

- **Unlearning in Large-Scale Foundation Models**: With the rise of Large Language Models (LLMs) in recommendation, there is a need to adapt our unlearning strategies to models with billions of parameters. Future studies will focus on "zero-shot unlearning" and parameter-efficient fine-tuning techniques to handle the "right to be forgotten" in massive, pre-trained architectures.

- **Cross-Domain Trustworthiness**: We plan to explore how the principles of noise robustness and unlearning can be transferred across different domains (e.g., from e-commerce to healthcare recommendations), ensuring that trustworthiness is maintained even when data distributions shift across platforms.

BIBLIOGRAPHY

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016).

Deep learning with differential privacy.

In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Abel, F., Herder, E., Houben, G.-J., Henze, N., and Krause, D. (2013).

Cross-system user modeling and personalization on the social web.

User Modeling and User-Adapted Interaction, 23:169–209.

Adomavicius, G. and Tuzhilin, A. (2005).

Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.

IEEE transactions on knowledge and data engineering, 17(6):734–749.

Aggarwal, C. C. (2016).

Knowledge-based recommender systems.

Recommender Systems: The Textbook, pages 167–197.

Akoglu, L., Tong, H., and Koutra, D. (2015).

Graph based anomaly detection and description: a survey.

Data mining and knowledge discovery, 29:626–688.

Ando, R. K., Zhang, T., and Bartlett, P. (2005).

A framework for learning predictive structures from multiple tasks and unlabeled data.

Journal of Machine Learning Research, 6(11).

Anelli, V. W., Deldjoo, Y., Di Noia, T., Malitesta, D., Paparella, V., and Pomo, C. (2023).

Auditing consumer-and producer-fairness in graph collaborative filtering.

In *ECIR*, volume 23.

Bai, J., Zhou, C., Song, J., Qu, X., An, W., Li, Z., and Gao, J. (2019).

Personalized bundle list recommendation.

In *The World Wide Web Conference*, pages 60–71.

Bai, T., Luo, J., Zhao, J., Wen, B., and Wang, Q. (2021a).

Recent advances in adversarial training for adversarial robustness.

arXiv preprint arXiv:2102.01356.

Bai, Y., Yang, E., Han, B., Yang, Y., Li, J., Mao, Y., Niu, G., and Liu, T. (2021b).

Understanding and improving early stopping for learning with noisy labels.

Advances in Neural Information Processing Systems, 34:24392–24403.

Barron, J. T. (2019).

A general and adaptive robust loss function.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4331–4339.

Baxter, J. (2000).

A model of inductive bias learning.

Journal of artificial intelligence research, 12:149–198.

Bennett, J., Lanning, S., et al. (2007).

The netflix prize.

In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York.

Bishop, C. M. and Nasrabadi, N. M. (2006).

Pattern recognition and machine learning, volume 4.

Springer.

Bouraga, S., Jureta, I., Faulkner, S., and Herssens, C. (2014).

Knowledge-based recommendation systems: A survey.

International Journal of Intelligent Information Technologies (IJIIT), 10(2):1–19.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. (2021).

Machine unlearning.

In *SP*, pages 141–159. IEEE.

Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D. (2017).

Unsupervised pixel-level domain adaptation with generative adversarial networks.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731.

Bradley, K., Rafter, R., and Smyth, B. (2000).

Case-based user profiling for content personalisation.

In *Adaptive Hypermedia and Adaptive Web-Based Systems: International Conference, AH 2000 Trento, Italy, August 28–30, 2000 Proceedings 1*, pages 62–72. Springer.

Bridge, D., Göker, M. H., McGinty, L., and Smyth, B. (2005).

Case-based recommender systems.

The Knowledge Engineering Review, 20(3):315–320.

Brophy, J. and Lowd, D. (2021).

Machine unlearning for random forests.

In *International Conference on Machine Learning*, pages 1092–1104. PMLR.

Burke, R. (2000).

Knowledge-based recommender systems.

Encyclopedia of library and information systems, 69(Supplement 32):175–186.

Burke, R. (2002).

Hybrid recommender systems: Survey and experiments.

User modeling and user-adapted interaction, 12:331–370.

Burke, R., O'Mahony, M. P., and Hurley, N. J. (2015).

Robust collaborative recommendation.

Recommender systems handbook, pages 961–995.

Camacho, L. A. G. and Alves-Souza, S. N. (2018).

Social network data to alleviate cold-start in recommender system: A systematic review.

Information Processing & Management, 54(4):529–544.

Çano, E. and Morisio, M. (2017).

Hybrid recommender systems: A systematic literature review.

Intelligent Data Analysis, 21(6):1487–1524.

Cao, Y. and Yang, J. (2015).

Towards making systems forget with machine unlearning.

In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., and Wallace, E. (2023).

Extracting training data from diffusion models.

arXiv preprint arXiv:2301.13188.

Cha, S., Cho, S., Hwang, D., Lee, H., Moon, T., and Lee, M. (2023).
Learning to unlearn: Instance-wise unlearning for pre-trained classifiers.
arXiv:2301.11578.

Cha, S., Cho, S., Hwang, D., Lee, H., Moon, T., and Lee, M. (2024).
Learning to unlearn: Instance-wise unlearning for pre-trained classifiers.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11186–11194.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. (2018).
Adversarial attacks and defences: A survey.
arXiv preprint arXiv:1810.00069.

Chan, A., Gujarati, A., Pattabiraman, K., and Gopalakrishnan, S.
Hierarchical unlearning framework for multi-class classification.
In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*.

Chen, C., Sun, F., Zhang, M., and Ding, B. (2022).
Recommendation unlearning.
In *WWW*, pages 2768–2777.

Chen, C., Zhang, M., Zhang, Y., Liu, Y., and Ma, S. (2020).
Efficient neural matrix factorization without sampling for recommendation.
TOIS, 38(2):1–28.

Chen, M., Gao, W., Liu, G., Peng, K., and Wang, C. (2023).
Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7766–7775.

Chen, P.-L., Tsai, C.-T., Chen, Y.-N., Chou, K.-C., Li, C.-L., Tsai, C.-H., Wu, K.-W., Chou, Y.-C., Li, C.-Y., Lin, W.-S., et al. (2012).

A linear ensemble of individual and blended models for music rating prediction.

In *Proceedings of KDD Cup 2011*, pages 21–60. PMLR.

Chen, W., Cai, F., Chen, H., and Rijke, M. D. (2019).

Joint neural collaborative filtering for recommender systems.

ACM Transactions on Information Systems (TOIS), 37(4):1–30.

Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. (2022).

Zero-shot machine unlearning.

arXiv preprint arXiv:2201.05629.

Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. (2023).

Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7210–7217.

Claypool, M., Brown, D., Le, P., and Waseda, M. (2001).

Inferring user interest.

IEEE Internet Computing, 5(6):32–39.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999).

Combing content-based and collaborative filters in an online newspaper.

In *Proc. of Workshop on Recommender Systems-Implementation and Evaluation*.

Collins, L., Mokhtari, A., and Shakkottai, S. (2020).

Task-robust model-agnostic meta-learning.

Advances in Neural Information Processing Systems, 33:18860–18871.

Colombo-Mendoza, L. O., Valencia-García, R., Rodríguez-González, A., Alor-Hernández, G., and Samper-Zapater, J. J. (2015).

Recommetz: A context-aware knowledge-based mobile recommender system for movie showtimes.

Expert Systems with Applications, 42(3):1202–1222.

Cour, T., Sapp, B., and Taskar, B. (2011).

Learning from partial labels.

The Journal of Machine Learning Research, 12:1501–1536.

Cui, L., Huang, W., Yan, Q., Yu, F. R., Wen, Z., and Lu, N. (2018).

A novel context-aware recommendation algorithm with two-level svd in social networks.

Future Generation Computer Systems, 86:1459–1470.

Cui, Z., Ma, J., Zhou, C., Zhou, J., and Yang, H. (2022).

M6-rec: Generative pretrained language models are open-ended recommender systems.

arXiv preprint arXiv:2205.08084.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B., et al. (2010).

The youtube video recommendation system.

In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296.

Del Olmo, F. H. and Gaudioso, E. (2008).

Evaluation of recommender systems: A new approach.

Expert Systems with Applications, 35(3):790–804.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018).

Bert: Pre-training of deep bidirectional transformers for language understanding.

arXiv:1810.04805.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019).

Bert: Pre-training of deep bidirectional transformers for language understanding.

In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Dezfouli, A., Nock, R., and Dayan, P. (2020).

Adversarial vulnerabilities of human decision-making.

Proceedings of the National Academy of Sciences, 117(46):29221–29228.

Di, Z., Zhu, Z., Jia, J., Liu, J., Takhirov, Z., Jiang, B., Yao, Y., Liu, S., and Liu, Y. (2024).

Label smoothing improves machine unlearning.

arXiv preprint arXiv:2406.07698.

Di Noia, T., Malitesta, D., and Merra, F. A. (2020).

Taamr: Targeted adversarial attack against multimedia recommender systems.

In *2020 50th Annual IEEE/IFIP international conference on dependable systems and networks workshops (DSN-W)*, pages 1–8. IEEE.

Dixit, V. S. and Jain, P. (2018).

An improved similarity measure to alleviate sparsity problem in context-aware recommender systems.

Towards Extensible and Adaptable Methods in Computing, pages 281–295.

Dong, J., Fang, Z., Liu, A., Sun, G., and Liu, T. (2021).

Confident anchor-induced multi-source free domain adaptation.

Advances in Neural Information Processing Systems, 34:2848–2860.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020).
An image is worth 16x16 words: Transformers for image recognition at scale.
arXiv:2010.11929.

Du, S. S., Koushik, J., Singh, A., and Póczos, B. (2017).
Hypothesis transfer learning via transformation functions.
Advances in neural information processing systems, 30.

Dwork, C. (2006).
Differential privacy.
In *International colloquium on automata, languages, and programming*, pages 1–12. Springer.

Ebrahimi, S., Meier, F., Calandra, R., Darrell, T., and Rohrbach, M. (2020).
Adversarial continual learning.
In *European Conference on Computer Vision*, pages 386–402. Springer.

Ekstrand, M. D., Riedl, J. T., Konstan, J. A., et al. (2011).
Collaborative filtering recommender systems.
Foundations and Trends® in Human–Computer Interaction, 4(2):81–173.

Erhan, D., Courville, A., Bengio, Y., and Vincent, P. (2010).
Why does unsupervised pre-training help deep learning?
In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. (2019).
Graph neural networks for social recommendation.
In *WWW*, pages 417–426.

Fan, Z., Liu, Z., Wang, Y., Wang, A., Nazari, Z., Zheng, L., Peng, H., and Yu, P. S. (2022).
Sequential recommendation via stochastic self-attention.
In *WWW*, pages 2036–2047.

Fatras, K., Damodaran, B. B., Lobry, S., Flamary, R., Tuia, D., and Courty, N. (2021).
Wasserstein adversarial regularization for learning with label noise.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(10):7296–7306.

Fei, K., Zhang, X., and Li, J. (2025).
Entire-space variational information exploitation for post-click conversion rate prediction.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 11654–11662.

Felfernig, A. and Burke, R. (2008).
Constraint-based recommender systems: technologies and research issues.
In *Proceedings of the 10th international conference on Electronic commerce*, pages 1–10.

Feng, L., Lv, J., Han, B., Xu, M., Niu, G., Geng, X., An, B., and Sugiyama, M. (2020).
Provably consistent partial-label learning.
Advances in neural information processing systems, 33:10948–10960.

Foster, J., Schoepf, S., and Brintrup, A. (2024).
Fast machine unlearning without retraining through selective synaptic dampening.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12043–12051.

Ganin, Y. and Lempitsky, V. (2015).
Unsupervised domain adaptation by backpropagation.

In *International conference on machine learning*, pages 1180–1189. PMLR.

Ge, Y., Liu, S., Fu, Z., Tan, J., Li, Z., Xu, S., Li, Y., Xian, Y., and Zhang, Y. (2022).
A survey on trustworthy recommender systems.
arXiv preprint arXiv:2207.12515.

Ghosh, A., Manwani, N., and Sastry, P. (2015).
Making risk minimization tolerant to label noise.
Neurocomputing, 160:93–107.

Ginart, A., Guan, M., Valiant, G., and Zou, J. Y. (2019).
Making ai forget you: Data deletion in machine learning.
NeurIPS, 32.

Golatkar, A., Achille, A., and Soatto, S. (2020).
Eternal sunshine of the spotless net: Selective forgetting in deep networks.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992).
Using collaborative filtering to weave an information tapestry.
Communications of the ACM, 35(12):61–70.

Goldberger, J. and Ben-Reuven, E. (2017).
Training deep neural-networks using a noise adaptation layer.
In *ICLR*.

Golub, G. H., Hansen, P. C., and O'Leary, D. P. (1999).
Tikhonov regularization and total least squares.
SIAM journal on matrix analysis and applications, 21(1):185–194.

Gong, M., Zhang, K., Liu, T., Tao, D., Glymour, C., and Schölkopf, B. (2016).

Domain adaptation with conditional transferable components.

In *International conference on machine learning*, pages 2839–2848. PMLR.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020).

Generative adversarial networks.

Communications of the ACM, 63(11):139–144.

Graves, L., Nagisetty, V., and Ganesh, V. (2021).

Amnesiac machine learning.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. (2009).

Covariate shift by kernel mean matching, dataset shift in machine learning, 131-160.

Guan, N., Liu, T., Zhang, Y., Tao, D., and Davis, L. S. (2017).

Truncated cauchy non-negative matrix factorization.

IEEE Transactions on pattern analysis and machine intelligence, 41(1):246–259.

Gui, J., Chen, T., Zhang, J., Cao, Q., Sun, Z., Luo, H., and Tao, D. (2024).

A survey on self-supervised learning: Algorithms, applications, and future trends.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12):9052–9071.

Gunawardana, A. and Meek, C. (2009).

A unified approach to building hybrid recommender systems.

In *Proceedings of the third ACM conference on Recommender systems*, pages 117–124.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017).

Deepfm: a factorization-machine based neural network for ctr prediction.
arXiv preprint arXiv:1703.04247.

Hampel, F. R. (1974).
The influence curve and its role in robust estimation.
Journal of the american statistical association, 69(346):383–393.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. (2018).
Co-teaching: Robust training of deep neural networks with extremely noisy labels.
Advances in neural information processing systems, 31.

Han, L., Luo, N., Huang, H., Chen, J., and Hartley, M.-A. (2024).
Towards independence criterion in machine unlearning of features and labels.
arXiv preprint arXiv:2403.08124.

He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, R. and McAuley, J. (2016a).
Fusing similarity models with markov chains for sparse sequential recommendation.
In *ICDM*, pages 191–200. IEEE.

He, R. and McAuley, J. (2016b).
Vbpr: visual bayesian personalized ranking from implicit feedback.
In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

He, X., Du, X., Wang, X., Tian, F., Tang, J., and Chua, T.-S. (2018).
Outer product-based neural collaborative filtering.
In *IJCAI*.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017).
Neural collaborative filtering.
In *WWW*, pages 173–182.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2015).
Session-based recommendations with recurrent neural networks.
arXiv:1511.06939.

Hinterstoisser, S., Lepetit, V., Wohlhart, P., and Konolige, K. (2018).
On pre-trained image features and synthetic images for deep learning.
In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Hofmann, T. (2004).
Latent semantic models for collaborative filtering.
ACM Transactions on Information Systems (TOIS), 22(1):89–115.

Hsieh, C.-K., Yang, L., Cui, Y., Lin, T.-Y., Belongie, S., and Estrin, D. (2017).
Collaborative metric learning.
In *WWW*, pages 193–201.

Huang, J., Zhang, L., Wang, J., Jiang, S., Huang, D., Ding, C., and Xu, L. (2024a).
Utilizing non-click samples via semi-supervised learning for conversion rate prediction.
In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 350–359.

Huang, M. H., Foo, L. G., and Liu, J. (2024b).
Learning to unlearn for robust machine unlearning.
arXiv preprint arXiv:2407.10494.

Huang, W., Ye, M., Shi, Z., Wan, G., Li, H., Du, B., and Yang, Q. (2024c).
Federated learning for generalization, robustness, fairness: A survey and benchmark.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12):9387–9406.

Hurley, N. J. (2011).

Robustness of recommender systems.

In *Proceedings of the fifth ACM conference on Recommender systems*, pages 9–10.

Izzo, Z., Smart, M. A., Chaudhuri, K., and Zou, J. (2021).

Approximate data deletion from machine learning models.

In *AISTATS*, pages 2008–2016. PMLR.

Jafri, S. I. H., Ghazali, R., Javid, I., Mahmood, Z., and Hassan, A. A. A. (2022).

Deep transfer learning with multimodal embedding to tackle cold-start and sparsity issues in recommendation system.

Plos one, 17(8):e0273486.

Jallouli, M., Lajmi, S., and Amous, I. (2017).

Designing recommender system: conceptual framework and practical implementation.

Procedia computer science, 112:1701–1710.

Jia, J., Liu, J., Ram, P., Yao, Y., Liu, G., Liu, Y., Sharma, P., and Liu, S. (2023).

Model sparsification can simplify machine unlearning.

arXiv preprint arXiv:2304.04934.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018).

MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels.

In *ICML*, pages 2309–2318.

Jiang, M., Cui, P., Chen, X., Wang, F., Zhu, W., and Yang, S. (2015).

Social recommendation with cross-domain transferable knowledge.

IEEE transactions on knowledge and data engineering, 27(11):3084–3097.

Jiang, M., Cui, P., and Faloutsos, C. (2016).

Suspicious behavior detection: Current trends and future directions.

IEEE intelligent systems, 31(1):31–39.

Jiang, Z., Zhou, K., Liu, Z., Li, L., Chen, R., Choi, S.-H., and Hu, X. (2022).

An information fusion approach to learning with instance-dependent label noise.

In *ICLR*.

Kang, W.-C. and McAuley, J. (2018).

Self-attentive sequential recommendation.

In *ICDM*, pages 197–206.

Kaur, D., Uslu, S., Rittichier, K. J., and Durresi, A. (2022).

Trustworthy artificial intelligence: a review.

ACM computing surveys (CSUR), 55(2):1–38.

Khan, Z. A., Chaudhary, N. I., and Zubair, S. (2019).

Fractional stochastic gradient descent for recommender systems.

Electronic Markets, 29:275–285.

Kim, J. and Woo, S. S. (2022).

Efficient two-stage model retraining for machine unlearning.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4361–4369.

Kingma, D. P. and Ba, J. (2014).

Adam: A method for stochastic optimization.

arXiv preprint arXiv:1412.6980.

Koren, Y., Bell, R., and Volinsky, C. (2009).

Matrix factorization techniques for recommender systems.

Computer, 42(8):30–37.

Koren, Y., Rendle, S., and Bell, R. (2021).

Advances in collaborative filtering.

Recommender systems handbook, pages 91–142.

Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T.-W., and Sejnowski, T. J. (2003).

Dictionary learning algorithms for sparse representation.

Neural computation, 15(2):349–396.

Krichene, W. and Rendle, S. (2020).

On sampled metrics for item recommendation.

In *SIGKDD*, pages 1748–1757.

Krizhevsky, A., Hinton, G., et al. (2009).

Learning multiple layers of features from tiny images.

Technical report.

Krohn-Grimberghe, A., Drumond, L., Freudenthaler, C., and Schmidt-Thieme, L. (2012).

Multi-relational matrix factorization using bayesian personalized ranking for social network data.

In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 173–182.

Kulkarni, S. and Rodd, S. F. (2020).

Context aware recommendation systems: A review of the state of the art techniques.

Computer Science Review, 37:100255.

Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., and Subrahmanian, V. (2018).

Rev2: Fraudulent user prediction in rating platforms.

In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. (2024).

Towards unbounded machine unlearning.

Advances in neural information processing systems, 36.

Le, Y. and Yang, X. (2015).

Tiny imagenet visual recognition challenge.

CS 231N, 7(7):3.

Lee, S. and Woo, S. S. (2023).

Undo: Effective and accurate unlearning method for deep neural networks.

In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4043–4047.

Lerche, L. and Jannach, D. (2014).

Using graded implicit feedback for bayesian personalized ranking.

In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 353–356.

Li, B., Yang, Q., and Xue, X. (2009).

Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction.

In *Twenty-First international joint conference on artificial intelligence*.

Li, C.-Y. and Lin, S.-D. (2014).

Matching users and items across domains to improve the recommendation quality.

In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 801–810.

Li, G., Shen, L., Sun, Y., Hu, Y., Hu, H., and Tao, D. (2023a).

Subspace based federated unlearning.

arXiv preprint arXiv:2302.12448.

Li, M., Soltanolkotabi, M., and Oymak, S. (2020).

Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks.

In *International conference on artificial intelligence and statistics*, pages 4313–4324. PMLR.

Li, S., Song, S.-j., and Wu, C. (2018).

Layer-wise domain correction for unsupervised domain adaptation.

Frontiers of Information Technology & Electronic Engineering, 19(1):91–103.

Li, Y., Chen, C., Zheng, X., Zhang, Y., Gong, B., and Wang, J. (2023b).

Selective and collaborative influence function for efficient recommendation unlearning.

arXiv:2304.10199.

Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., and Ma, X. (2021).

Anti-backdoor learning: Training clean models on poisoned data.

Advances in Neural Information Processing Systems, 34:14900–14912.

Li, Y., Zheng, X., Chen, C., and Liu, J. (2022a).

Making recommender systems forget: Learning and unlearning for erasable recommendation.

arXiv:2203.11491.

Li, Z., Ji, J., Ge, Y., and Zhang, Y. (2022b).

Autolossgen: Automatic loss function generation for recommender systems.

In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1304–1315.

Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. (2018).

Variational autoencoders for collaborative filtering.

In *WWW*, pages 689–698.

Liang, F., Pan, W., and Ming, Z. (2021).

Fedrec++: Lossless federated recommendation with explicit feedback.

In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4224–4231.

Lin, Z., Tian, C., Hou, Y., and Zhao, W. X. (2022).

Improving graph collaborative filtering with neighborhood-enriched contrastive learning.

In *WWW*.

Linden, G., Smith, B., and York, J. (2003).

Amazon. com recommendations: Item-to-item collaborative filtering.

IEEE Internet computing, 7(1):76–80.

Liu, T. and Tao, D. (2015).

Classification with noisy labels by importance reweighting.

IEEE Transactions on pattern analysis and machine intelligence, 38(3):447–461.

Liu, W., Wan, J., Wang, X., Zhang, W., Zhang, D., and Li, H. (2022).

Forgetting fast in recommender systems.

arXiv:2208.06875.

Liu, Z., Chen, Y., Li, J., Yu, P. S., McAuley, J., and Xiong, C. (2021).

Contrastive self-supervised sequential recommendation with robust augmentation.

arXiv preprint arXiv:2108.06479.

Loni, B., Pagano, R., Larson, M., and Hanjalic, A. (2016).

Bayesian personalized ranking with multi-channel user feedback.
In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 361–364.

Lops, P., De Gemmis, M., and Semeraro, G. (2011).
Content-based recommender systems: State of the art and trends.
Recommender systems handbook, pages 73–105.

Lorenzi, F. and Ricci, F. (2003).
Case-based recommender systems: A unifying view.
In *IJCAI Workshop on Intelligent Techniques for Web Personalization*, pages 89–113.
Springer.

Lu, J. (2004).
A personalized e-learning material recommender system.
In *International conference on information technology and applications*. Macquarie Scientific Publishing.

Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. (2015).
Recommender system application developments: a survey.
Decision support systems, 74:12–32.

Lu, J., Zhang, Q., and Zhang, G. (2020).
Recommender Systems: Advanced Developments.
World Scientific.

Lu, K., Lu, J., Xu, H., Guo, K., Zhang, Q., Lin, H., Grosser, M., Zhang, Y., and Zhang, G. (2025).
Genomics-enhanced cancer risk prediction for personalized llms-driven healthcare recommender systems.
ACM Transactions on Information Systems.

Lu, K., Zhang, Q., Hughes, D., Zhang, G., and Lu, J. (2024).

Amt-cdr: A deep adversarial multi-channel transfer network for cross-domain recommendation.

ACM Transactions on Intelligent Systems and Technology, 15(4):1–26.

Luo, X., Zhou, M., Li, S., You, Z., Xia, Y., and Zhu, Q. (2015).

A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method.

IEEE transactions on neural networks and learning systems, 27(3):579–592.

Lv, J., Xu, M., Feng, L., Niu, G., Geng, X., and Sugiyama, M. (2020).

Progressive identification of true labels for partial-label learning.

In *international conference on machine learning*, pages 6500–6510. PMLR.

Ma, C., Kang, P., and Liu, X. (2019).

Hierarchical gating networks for sequential recommendation.

In *SIGKDD*, pages 825–833.

Ma, J., Zhao, Z., Yi, X., Chen, J., Hong, L., and Chi, E. H. (2018a).

Modeling task relationships in multi-task learning with multi-gate mixture-of-experts.

In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1930–1939.

Ma, X., Zhao, L., Huang, G., Wang, Z., Hu, Z., Zhu, X., and Gai, K. (2018b).

Entire space multi-task model: An effective approach for estimating post-click conversion rate.

In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1137–1140.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017).

Towards deep learning models resistant to adversarial attacks.

arXiv preprint arXiv:1706.06083.

Mahadevan, A. and Mathioudakis, M. (2021).

Certifiable machine unlearning for linear models.

arXiv preprint arXiv:2106.15093.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Hung Byers, A., et al. (2011).

Big data: The next frontier for innovation, competition, and productivity.

McKinsey Global Institute.

Mehta, B., Hofmann, T., and Nejdl, W. (2007).

Robust collaborative filtering.

In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 49–56.

Mehta, R., Pal, S., Singh, V., and Ravi, S. N. (2022).

Deep unlearning via randomized conditionally independent hessians.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10422–10431.

Mehta, R. and Rana, K. (2017).

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.

Menon, A. K., Rawat, A. S., Reddi, S. J., and Kumar, S. (2020).

Can gradient clipping mitigate label noise?

In *International Conference on Learning Representations*.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018).

Foundations of machine learning.

MIT press.

Muandet, K., Balduzzi, D., and Schölkopf, B. (2013).

Domain generalization via invariant feature representation.

In *International conference on machine learning*, pages 10–18. PMLR.

Murphy, K. P. (2012).

Machine learning: a probabilistic perspective.

MIT press.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2013).

Learning with noisy labels.

Advances in neural information processing systems, 26.

Natarajan, S., Vairavasundaram, S., Natarajan, S., and Gandomi, A. H. (2020).

Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data.

Expert Systems with Applications, 149:113248.

Nguyen, T. T. S., Lu, H. Y., and Lu, J. (2013).

Web-page recommendation based on web usage and domain knowledge.

IEEE Transactions on Knowledge and Data Engineering, 26(10):2574–2587.

Nilashi, M., Bagherifard, K., Ibrahim, O., Alizadeh, H., Nojeem, L. A., and Roozegar, N. (2013).

Collaborative filtering recommender systems.

Research Journal of Applied Sciences, Engineering and Technology, 5(16):4168–4182.

Núñez-Valdez, E. R., Quintana, D., Crespo, R. G., Isasi, P., and Herrera-Viedma, E. (2018).

A recommender system based on implicit feedback for selective dissemination of ebooks.

Information Sciences, 467:87–98.

Oard, D. W., Kim, J., et al. (1998).

Implicit feedback for recommender systems.

In *Proceedings of the AAAI workshop on recommender systems*, volume 83, pages 81–83. Madison, WI.

Ouyang, W., Dong, R., Zhang, X., Guo, C., Luo, J., Liu, X., and Du, Y. (2023).

Contrastive learning for conversion rate prediction.

In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1909–1913.

Pan, S. J. and Yang, Q. (2010).

A survey on transfer learning.

IEEE Transactions on knowledge and data engineering, 22(10):1345–1359.

Paterek, A. (2007).

Improving regularized singular value decomposition for collaborative filtering.

In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8.

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. (2017).

Making deep neural networks robust to label noise: A loss correction approach.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952.

Pentina, A. and Lampert, C. (2014).

A pac-bayesian bound for lifelong learning.

In *International Conference on Machine Learning*, pages 991–999. PMLR.

Poppi, S., Sarto, S., Cornia, M., Baraldi, L., and Cucchiara, R. (2024).
Multi-class unlearning for image classification via weight filtering.
IEEE Intelligent Systems.

Quadrana, M., Karatzoglou, A., Hidasi, B., and Cremonesi, P. (2017).
Personalizing session-based recommendations with hierarchical recurrent neural networks.
In *RecSys*, pages 130–137.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007).
Self-taught learning: transfer learning from unlabeled data.
In *Proceedings of the 24th international conference on Machine learning*, pages 759–766.

Rajendran, D. P. D. and Sundarraj, R. P. (2021).
Using topic models with browsing history in hybrid collaborative filtering recommender system: Experiments with user ratings.
International Journal of Information Management Data Insights, 1(2):100027.

Rayana, S. and Akoglu, L. (2015).
Collective opinion spam detection: Bridging review networks and metadata.
In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, pages 985–994.

Raza, S. and Ding, C. (2019).
Progress in context-aware recommender systems,Äîan overview.
Computer Science Review, 31:84–97.

Regulation, G. D. P. (2018).
General data protection regulation (gdpr).

Intersoft Consulting, Accessed in October, 24(1).

Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018).

Learning to reweight examples for robust deep learning.

In *International conference on machine learning*, pages 4334–4343. PMLR.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2012).

Bpr: Bayesian personalized ranking from implicit feedback.

arXiv preprint arXiv:1205.2618.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010).

Factorizing personalized markov chains for next-basket recommendation.

In *WWW*, pages 811–820.

Rendle, S. and Schmidt-Thieme, L. (2008).

Online-updating regularized kernel matrix factorization models for large-scale recommender systems.

In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258.

Resnick, P. and Varian, H. R. (1997).

Recommender systems.

Communications of the ACM, 40(3):56–58.

Rodpysh, K. V., Mirabedini, S. J., and Banirostam, T. (2021).

Resolving cold start and sparse data challenge in recommender systems using multi-level singular value decomposition.

Computers & Electrical Engineering, 94:107361.

Roy, D. and Dutta, M. (2022).

A systematic review and research perspective on recommender systems.

Journal of Big Data, 9(1):59.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000).

Application of dimensionality reduction in recommender system-a case study.

Technical report, Minnesota Univ Minneapolis Dept of Computer Science.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001).

Item-based collaborative filtering recommendation algorithms.

In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007).

Collaborative filtering recommender systems.

The adaptive web: methods and strategies of web personalization, pages 291–324.

Schafer, J. B., Konstan, J., and Riedl, J. (1999).

Recommender systems in e-commerce.

In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166.

Schelter, S., Grafberger, S., and Dunning, T. (2021).

Hedgecut: Maintaining randomised trees for low-latency machine unlearning.

In *SIGMOD*, pages 1545–1557.

Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., and Mooij, J. (2012).

On causal and anticausal learning.

arXiv preprint arXiv:1206.6471.

Scott, C. (2015).

A rate of convergence for mixture proportion estimation, with application to learning from noisy labels.

In *AISTATS*, pages 838–846.

Sekhari, A., Acharya, J., Kamath, G., and Suresh, A. T. (2021).

Remember what you want to forget: Algorithms for machine unlearning.

NeurIPS, 34:18075–18086.

Setlur, A., Eysenbach, B., Smith, V., and Levine, S. (2022).

Adversarial unlearning: Reducing confidence along adversarial directions.

Advances in Neural Information Processing Systems, 35:18556–18570.

Shambour, Q. and Lu, J. (2015).

An effective recommender system by unifying user and item trust information for b2b applications.

Journal of Computer and System Sciences, 81(7):1110–1126.

Shibata, T., Irie, G., Ikami, D., and Mitsuzumi, Y. (2021).

Learning with selective forgetting.

In *IJCAI*, volume 3, page 4.

Shu, J., Shen, X., Liu, H., Yi, B., and Zhang, Z. (2018).

A content-based recommendation algorithm for learning resources.

Multimedia Systems, 24(2):163–173.

Singh, A. P. and Gordon, G. J. (2008).

Relational learning via collective matrix factorization.

In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658.

Smyth, B. (2007).

Case-based recommendation.

The adaptive web: methods and strategies of web personalization, pages 342–376.

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. (2022).

Learning from noisy labels with deep neural networks: A survey.

IEEE Transactions on Neural Networks and Learning Systems.

Su, H., Meng, L., Zhu, L., Lu, K., and Li, J. (2024).

Ddpo: Direct dual propensity optimization for post-click conversion rate estimation.

In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1179–1188.

Sugiyama, M., Bao, H., Ishida, T., Lu, N., and Sakai, T. (2022).

Machine learning from weak supervision: An empirical risk minimization approach.

MIT Press.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. (2019a).

Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer.

In *CIKM*, pages 1441–1450.

Sun, J., Zhang, Y., Ma, C., Coates, M., Guo, H., Tang, R., and He, X. (2019b).

Multi-graph convolution collaborative filtering.

In *ICDM*, pages 1306–1311.

Suriyakumar, V. M. and Wilson, A. C. (2022).

Algorithms that approximate data removal: New results and limitations.

Tang, J., Du, X., He, X., Yuan, F., Tian, Q., and Chua, T.-S. (2019).

Adversarial training towards robust multimedia recommender system.

IEEE Transactions on Knowledge and Data Engineering, 32(5):855–867.

Tang, J. and Wang, K. (2018).

Personalized top-n sequential recommendation via convolutional sequence embedding.

In *WSDM*, pages 565–573.

Tarun, A. K., Chundawat, V. S., Mandal, M., and Kankanhalli, M. (2023).

Fast yet effective machine unlearning.

IEEE Transactions on Neural Networks and Learning Systems.

Terveen, L., Hill, W., et al. (2001).

Human-computer collaboration in recommender systems.

HCI in the New Millennium.

Torrey, L. and Shavlik, J. (2010).

Transfer learning.

In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Tsai, C.-H., Lin, C.-Y., and Lin, C.-J. (2014).

Incremental and decremental training for linear classification.

In *SIGKDD*, pages 343–352.

Vapnik, V. (2013).

The nature of statistical learning theory: Springer science & business media.

Berlin, Germany.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017).

Attention is all you need.

In *NeurIPS*, pages 5998–6008.

Villegas, N. M., Sánchez, C., Díaz-Cely, J., and Tamura, G. (2018).

Characterizing context-aware recommender systems: A systematic literature review.

Knowledge-Based Systems, 140:173–200.

Wang, H., Chang, T.-W., Liu, T., Huang, J., Chen, Z., Yu, C., Li, R., and Chu, W. (2022a).

Escm2: Entire space counterfactual multi-task model for post-click conversion rate estimation.

In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 363–372.

Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q., and Orgun, M. (2019).
Sequential recommender systems: Challenges, progress and prospects.
In *Twenty-Eighth International Joint Conference on Artificial Intelligence* {*IJCAI-19*}.
International Joint Conferences on Artificial Intelligence Organization.

Wang, S., Zhang, X., Wang, Y., Liu, H., and Ricci, F. (2022b).
Trustworthy recommender systems.
arXiv preprint arXiv:2208.06265.

Wang, W., Feng, F., He, X., Wang, X., and Chua, T.-S. (2021).
Deconfounded recommendation for alleviating bias amplification.
In *SIGKDD*, pages 1717–1725.

Wang, X. and Schneider, J. (2014).
Flexible transfer learning under support and model shift.
Advances in Neural Information Processing Systems, 27.

Wei, H., Tao, L., Xie, R., and An, B. (2021).
Open-set label noise can improve robustness against inherent label noise.
In *NeurIPS*, pages 7978–7992.

Wei, Y., Chen, S., Ye, S., Han, B., and Gong, C. (2025).
Robust learning under hybrid noise.
ACM Transactions on Intelligent Systems and Technology.

Wen, H., Zhang, J., Lin, Q., Yang, K., and Huang, P. (2019).
Multi-level deep cascade trees for conversion rate prediction in recommendation system.

In *proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 338–345.

Wen, H., Zhang, J., Wang, Y., Lv, F., Bao, W., Lin, Q., and Yang, K. (2020).
Entire space multi-task modeling via post-click behavior decomposition for conversion rate prediction.
In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 2377–2386.

Wu, C., Lian, D., Ge, Y., Zhu, Z., Chen, E., and Yuan, S. (2021a).
Fight fire with fire: towards robust recommender systems via adversarial poisoning training.
In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1074–1083.

Wu, G., Hashemi, M., and Srinivasa, C. (2022a).
Puma: Performance unchanged model augmentation for training data removal.
In *AAAI*, volume 36, pages 8675–8682.

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., and Xie, X. (2021b).
Self-supervised graph learning for recommendation.
In *SIGIR*, pages 726–735.

Wu, L., Guo, S., Wang, J., Hong, Z., Zhang, J., and Ding, Y. (2022b).
Federated unlearning: Guarantee the right of clients to forget.
IEEE Network, 36(5):129–135.

Wu, Y., DuBois, C., Zheng, A. X., and Ester, M. (2016).
Collaborative denoising auto-encoders for top-n recommender systems.
In *WSDM*, pages 153–162.

Xia, L., Huang, C., Xu, Y., Zhao, J., Yin, D., and Huang, J. (2022).

Hypergraph contrastive collaborative filtering.

In *SIGIR on Research and Development in Information Retrieval*, pages 70–79.

Xia, X., Liu, T., Han, B., Gong, C., Wang, N., Ge, Z., and Chang, Y. (2021).

Robust early-learning: Hindering the memorization of noisy labels.

In *International conference on learning representations*.

Xia, X., Liu, T., Wang, N., Han, B., Gong, C., Niu, G., and Sugiyama, M. (2019).

Are anchor points really indispensable in label-noise learning?

Advances in neural information processing systems, 32.

Xiao, H., Rasul, K., and Vollgraf, R. (2017).

Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

arXiv preprint arXiv:1708.07747.

Xiao, T. and Wang, S. (2022).

Towards unbiased and robust causal ranking for recommender systems.

In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1158–1167.

Xie, X., Sun, F., Liu, Z., Wu, S., Gao, J., Zhang, J., Ding, B., and Cui, B. (2022).

Contrastive learning for sequential recommendation.

In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 1259–1273. IEEE.

Xu, M., Lian, Z., Feng, L., Liu, B., and Tao, J. (2023a).

Alim: adjusting label importance mechanism for noisy partial label learning.

Advances in Neural Information Processing Systems, 36:38668–38684.

Xu, M., Sun, J., Yang, X., Yao, K., and Wang, C. (2023b).

Netflix and forget: Efficient and exact machine unlearning from bi-linear recommendations.

arXiv:2302.06676.

Xu, N., Lv, J., and Geng, X. (2019).

Partial label learning via label enhancement.

In *Proceedings of the AAAI Conference on artificial intelligence*, volume 33, pages 5557–5564.

Xu, N., Qiao, C., Geng, X., and Zhang, M.-L. (2021a).

Instance-dependent partial label learning.

Advances in Neural Information Processing Systems, 34:27119–27130.

Xu, S., Tan, J., Heinecke, S., Li, V. J., and Zhang, Y. (2021b).

Deconfounded causal collaborative filtering.

ACM Transactions on Recommender Systems.

Yan, H., Li, X., Guo, Z., Li, H., Li, F., and Lin, X. (2022).

Arcane: An efficient architecture for exact machine unlearning.

In *IJCAI*, pages 4006–4013.

Yang, Z., Xu, L., Cai, Z., and Xu, Z. (2016).

Re-scale adaboost for attack detection in collaborative filtering recommender systems.

Knowledge-Based Systems, 100:74–88.

Yao, L., Sheng, Q. Z., Segev, A., and Yu, J. (2013).

Recommending web services via combining collaborative filtering with content-based features.

In *2013 IEEE 20th International Conference on Web Services*, pages 42–49. IEEE.

Yao, Y., Liu, T., Han, B., Gong, M., Deng, J., Niu, G., and Sugiyama, M. (2020).

Dual t: Reducing estimation error for transition matrix in label-noise learning.

Advances in neural information processing systems, 33:7260–7271.

Ye, J., Fu, Y., Song, J., Yang, X., Liu, S., Jin, X., Song, M., and Wang, X. (2022).

Learning with recoverable forgetting.

arXiv preprint arXiv:2207.08224.

Ye, S. and Lu, J. (2023).

Sequence unlearning for sequential recommender systems.

In *Australasian Joint Conference on Artificial Intelligence*, pages 403–415.

Ye, S. and Lu, J. (2024).

Robust recommender systems with rating flip noise.

ACM Transactions on Intelligent Systems and Technology.

Ye, S., Lu, J., and Zhang, G. (2025).

Towards safe machine unlearning: A paradigm that mitigates performance degradation.

In *Proceedings of the ACM on Web Conference 2025*, pages 4635–4652.

Yoon, Y., Nam, J., Yun, H., Kim, D., and Ok, J. (2022).

Few-shot unlearning by model inversion.

arXiv preprint arXiv:2205.15567.

Yu, D., Li, Q., Wang, X., and Xu, G. (2023).

Deconfounded recommendation via causal intervention.

Neurocomputing, 529:128–139.

Yu, S., Sun, F., Guo, J., Zhang, R., and Cheng, X. (2022).

Legonet: A fast and exact unlearning architecture.

arXiv preprint arXiv:2210.16023.

Yuan, W., Nguyen, Q. V. H., He, T., Chen, L., and Yin, H. (2023a).

Manipulating federated recommender systems: Poisoning with synthetic users and its countermeasures.

arXiv preprint arXiv:2304.03054.

Yuan, W., Yin, H., Wu, F., Zhang, S., He, T., and Wang, H. (2023b).

Federated unlearning for on-device recommendation.

In *WSDM*, pages 393–401.

Yuan, W., Yuan, S., Zheng, K., Nguyen, Q. V. H., and Yin, H. (2023c).

Manipulating visually-aware federated recommender systems and its countermeasures.

arXiv preprint arXiv:2305.08183.

Zangerle, E. and Bauer, C. (2022).

Evaluating recommender systems: Survey and framework.

ACM Computing Surveys, 55(8):1–38.

Zhang, F., Lu, Y., Chen, J., Liu, S., and Ling, Z. (2017).

Robust collaborative filtering based on non-negative matrix factorization and r1-norm.

Knowledge-based systems, 118:177–190.

Zhang, K., Schölkopf, B., Muandet, K., and Wang, Z. (2013).

Domain adaptation under target and conditional shift.

In *International conference on machine learning*, pages 819–827. PMLR.

Zhang, L. (2013).

The definition of novelty in recommendation system.

Journal of Engineering Science & Technology Review, 6(3).

Zhang, M.-L., Zhou, B.-B., and Liu, X.-Y. (2016).

Partial label learning via feature-aware disambiguation.

In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1335–1344.

Zhang, Q. (2018).

Enhanced recommender systems through cross-domain knowledge transfer.

PhD thesis.

Zhang, Q., Lu, J., and Jin, Y. (2021a).

Artificial intelligence in recommender systems.

Complex & Intelligent Systems, 7:439–457.

Zhang, Y., Hu, Z., Bai, Y., Feng, F., Wu, J., Wang, Q., and He, X. (2023).

Recommendation unlearning via influence function.

arXiv:2307.02147.

Zhang, Y., Niu, G., and Sugiyama, M. (2021b).

Learning noise transition matrix from only noisy labels via total variation regularization.

In *ICML*, pages 12501–12512.

Zhang, Y., Tan, Y., Zhang, M., Liu, Y., Chua, T.-S., and Ma, S. (2015).

Catch the black sheep: unified framework for shilling attack detection based on fraudulent action propagation.

In *IJCAI*.

Zhou, X., Liu, X., Jiang, J., Gao, X., and Ji, X. (2021).

Asymmetric loss functions for learning with noisy labels.

In *ICML*, pages 12846–12856.

Zhou, Z.-H. (2018).

A brief introduction to weakly supervised learning.

National science review, 5(1):44–53.

Zhu, F., Zhong, M., Yang, X., Li, L., Yu, L., Zhang, T., Zhou, J., Chen, C., Wu, F., Liu, G., et al. (2023).

Dcmt: a direct entire-space causal multi-task framework for post-click conversion estimation.

In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 3113–3125. IEEE.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020).

A comprehensive survey on transfer learning.

Proceedings of the IEEE, 109(1):43–76.