Non-IID Learning for Recommendation, Time Series and Hashing

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

by

Qi Zhang

under the supervision of

Prof. Longbing Cao

to

School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

May 2023

CERTIFICATE OF ORIGINAL OWNERSHIP

I, *Qi Zhang*, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *Faculty of Engineering and Information Technology* at the University of Technology Sydney.

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.

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with *Beijing Institute of Technology*.

This research is supported by the Australian Government Research Training Program.

SIGNATURE: _____

[Qi Zhang]

DATE: 5th December, 2022

PLACE: Sydney, Australia

Abstract

With the prevalence of information technology, a huge amount of data emerges every day from various domains and has been pervasive in our daily living, studying, working, and entertaining applications. In such a big data era, data learning playing a major role in transforming the thinking of data science has dominated research communities and business applications. Meanwhile, the increasing complexities of real-world data, e.g., heterogeneity and coupling relationships, extremely challenge the existing data learning methodologies and techniques and may seriously limit their applicability and feasibility.

For several decades, the **i**ndependent and **i**dentically **d**istributed (short for *IID*) assumption has laid the foundation of data learning, simplifying real-world data's intricate nature for effectively achieving approximate, traceability, and asymptotic problem-solving. Unfortunately, real-world scenarios generally go beyond the IID assumption and count on specific knowledge and capability to address practical problems and challenges, where IID may show significant limitations and gaps. A broad-reaching non-IID thinking is to explore and exploit the intrinsic heterogeneities and couplings of real-world data, which has been increasingly attractive and prevalent in data learning research and applications. However, non-IIDness shows diversified properties with different data scenarios, for example, heterogeneities in data types, attributes, sources, and couplings within and between structures, distributions, and variables. It is far from reaching a unified non-IID learning paradigm for addressing various real-world heterogeneities and couplings. More importantly, it is also extremely challenging to exhaustively tailor non-IID data learning methodologies for specified scenarios and applications.

In this thesis, I explore non-IID learning in terms of different applications, specifically recommender systems, multivariate time series (MTS) analysis, and learning to hash, to enlighten non-IID methodologies and techniques. The elaborately chosen applications and scenarios penetrate our daily living, studying, working, and entertaining activities, and cover various tasks of classification, ranking, representation, and retrieval. Accordingly, the main research objectives include modeling and learning the non-IIDness in recommender systems, multivariate time series (MTS) analysis, and learning to hash, respectively, and delivering non-IID models to effectively handle the scenarios with both IID and non-IID data.

- To build non-IID recommender systems, we make attempts from two aspects: 1) learning user/item/context feature couplings, and 2) modeling rating distribution heterogeneity. First, we analyze the user/item/context coupling relationships and

their influence on user actions; and then build a neural time-aware recommendation model with a specified feature interaction network to factorize the pairwise couplings between users, items, and temporal context. Second, we analyze the potential rating generation process which intrinsically determines the rating distribution heterogeneity. Accordingly, we propose a tripartite collaborative filtering framework and instantiate a tripartite probabilistic matrix factorization to model the rating generation and eliminate the distribution bias for debiasing rating estimation.

- To perform non-IID MTS forecasting, we jointly model inter- and intra-series coupling relationships and inter-series heterogeneities. Specifically, we first propose a non-IID MTS forecasting model integrating spectral clustering and Transformer. The model introduces a spectral clustering network that adaptively learns to segregate heterogeneous time series and a clusterwise forecasting network with multi-channel Transformers to model intra- and inter-series couplings. In addition, we revisit the coupling relationships from the perspective of mutual information and propose a deep coupling network that introduces a coupling module to explicitly model variable relationships and a coupling representation module to encode high-order coupling patterns.

- To model non-IID learning to hash, we aim to 1) preserve the couplings between inputs and hash codes, and 2) address the high-dimensional and heterogeneous issues. First, we study the impracticality of conventional code balance constraints and then introduce probabilistic code constraints to improve hash quality by guaranteeing the mutual informativeness between inputs and their hash codes. Second, we apply deep supervised hashing on high-dimensional and heterogeneous data and propose a deep hashing network to learn similarity-preserving hash codes for efficient case retrieval and deliver deep-hashing-enabled case-based reasoning. The network introduces position embedding to represent heterogeneous features and utilizes a multilinear interaction layer to effectively filtrate zero-valued features for addressing the sparsity issues and capturing feature couplings.

Thorough empirical evaluations have been conducted on real-world datasets to compare our proposed methods with the state-of-the-art approaches. The results prove that our non-IID modeling methods effectively address real-world couplings and heterogeneity issues in various complex data and significantly benefit the corresponding specific applications.

Dedication

To my family...

ACKNOWLEDGMENTS

Many thanks to my supervisor Prof. Longbing Cao for his comprehensive guidance and help in my Ph.D. study and research, and for his rigorousness, patience, wisdom, and profession. His continuous guidance and assistance helped me in my doctoral career and taught me research methodology, critical thinking, writing skills, and scientific knowledge. His advice on both my research and career is priceless and will help me in my academic career and life. I met Prof. Longbing Cao for the first time when I was pursuing my Ph.D. at the Beijing Institute of Technology. His strong knowledge, dedication, and motivation in his research left me with a deep impression from then. I have always been proud and fortunate to be a member of the Data Science Lab, University of Technology Sydney and to study with Prof. Longbing Cao.

I would also like to express my gratitude to Prof. Chongyang Shi and Prof. Zhendong Niu for their help when I was pursuing my first Ph.D. at the Beijing Institute of Technology. Especially during COVID-19, I spent a hard time at the Beijing Institute of Technology conducting my research and completing my thesis. They also provide me with useful suggestions about my career and research. Their strong sociability inspires me and helps me to start my career development.

Special thanks to my family and friends. Nothing can express my gratitude to my parents for your meticulous care of me and all of the sacrifices that you have made on my behalf. I know it was a hard time when I studied aboard and was far away from home. I want to express deep appreciation to you for being my strongest back support all the time. Moreover, I would like to appreciate my dear friends for accompanying me as my spiritual partners and for their encouragement and help to me. My thesis would not have been completed without their warm support and endless love.

Many thanks to my lab mates for their kind support and help. I appreciate the studying time with Liang Hu, Shoujin Wang, Chenzhang Zhu, Prof. Defu Lian, Longxiang Shi, Shufeng Hao, Guansong Pang, and Ke Liu, and appreciate the impressive research period I spent with them in Australia. In addition, I would thank my coauthors including Liang Hu, Shoujin Wang, Chengzhang Zhu, Longxiang Shi, and Xinyu Jiang. I am motivated and impressed by their hardworking, diligence, brilliance, and great motivation. I learned a lot about research and life from them.

List of Publications

Related Publications to the Thesis

1. **Qi Zhang**, Longbing Cao, Chongyang Shi, Zhendong Niu: Neural Time-Aware Sequential Recommendation by Jointly Modeling Preference Dynamics and Explicit Feature Couplings. IEEE Trans. Neural Networks Learn. Syst. 33(10): 5125-5137 (2022). (**Chapter 4**)

2. **Qi Zhang**, Longbing Cao, Chongyang Shi, Liang Hu: Tripartite Collaborative Filtering with Observability and Selection for Debiasing Rating Estimation on Missing-Not-at-Random Data. AAAI 2021: 4671-4678. (**Chapter 5**)

3. **Qi Zhang**, Liang Hu, Chongyang Shi, Shoujin Wang, Longbing Cao: Cospectrumer: Spectral Clustering-enhanced Transformer for Non-IID Multivariate Time Series. IEEE Trans. Pattern Anal. Mach. Intell. (Submitted, **Chapter 6**)

4. Deep Coupling Network For Multivariate Time Series Forecasting (ongoing, **Chapter 7**)

5. **Qi Zhang**, Liang Hu, Longbing Cao, Chongyang Shi, Shoujin Wang, Dora D. Liu: A Probabilistic Code Balance Constraint with Compactness and Informativeness Enhancement for Deep Supervised Hashing. IJCAI 2022: 1651-1657. (**Chapter 8**)

6. **Qi Zhang**, Liang Hu, Chongyang Shi, Ke Liu, Longbing Cao: Supervised Deep Hashing for High-dimensional and Heterogeneous Case-based Reasoning. ACM Trans. Inf. Syst. (Submitted, **Chapter 9**)

OTHERS PUBLICATIONS:

7. **Qi Zhang**, C. Shi, Z. Niu and L. Cao, HCBC: A Hierarchical Case-Based Classifier Integrated with Conceptual Clustering, IEEE Trans. Knowl. Data Eng., vol. 31, no. 1, pp. 152-165, 1 Jan. 2019.

8. **Qi Zhang***, Chengzhang Zhu*, Longbing Cao, Arman Abrahamyan. Mix2Vec: Unsupervised Mixed Data Representation, DSAA, Sydney, NSW, Australia, 2020, pp. 118-127.

9. **Qi Zhang**, Chongyang Shi, Ping Sun, and Zhengdong Niu. Case-based classification on hierarchical structure of formal concept analysis. ECAI. IOS Press, NLD, 1758-1759.

10. Liang Hu, Dora Liu, **Qi Zhang**, Tangwei Ye, Usman Naseem, Zhongyuan Lai. A Dynamics and Task Decoupled Reinforcement Learning Architecture for High-efficiency Dynamic Target Intercept, AAAI 2023.

11. Dora Liu, Liang Hu, **Qi Zhang**, Usman Naseem, Zhongyuan Lai, Self-supervised Learning Temporal Causalities behind Human Actions for Multilevel Skeleton Forgery Detection, AAAI 2023.

12. Hui He, **Qi Zhang**, Simeng Bai, Kun Yi, Zhendong Niu. CATN: Cross Attentive Tree-Aware Network For Multivariate Time Series Forecasting, AAAI, 2022.

13. Xinyu Jiang*, **Qi Zhang***, Chongyang Shi, Kaiying Jiang, Liang Hu, Shoujin Wang. An Ion-Exchange Mechanism Inspired Story Ending Generator for Different Characters. ECML-PKDD 2022.

14. Shoujin Wang, **Qi Zhang**, Liang Hu, Xiuzhen Zhang, Yan Wang, Charu Aggarwal. Sequential/Session-based Recommendations: Challenges, Approaches, Applications and Opportunities. SIGIR 2022.

15. Xinyu Jiang, **Qi Zhang**, Chongyang Shi. Hierarchical Neural Network with Bidirectional Selection Mechanism for Sentiment Analysis. IJCNN, 2022.

16. Chaoqun Feng, Chongyang Shi, Shufeng Hao, **Qi Zhang**, Xinyu Jiang, Daohua Yu. Hierarchical Social Similarity-guided Model with Dual-mode Attention for Session-based Recommendation, Knowl. Based Syst., Volume 176, 2021, 114834, ISSN 0957-4174.

17. Chaoqun Feng, Chongyang Shi, Chuanming Liu, **Qi Zhang**, Shufeng Hao, Xinyu Jiang. Context-aware item attraction model for session-based recommendation, Expert Syst. Appl., Volume 176, 2021, 114834, ISSN 0957-4174.

18. ZhiyongDai, Jianjun Yi, Lei Yan, Qingwen Xu, Liang Hu, **Qi Zhang**#, Jiahui Li, Guoqiang Wang. PFEMed: Few-shot medical image classification using prior guided feature enhancement, Pattern Recognit., Volume 134, February 2023, 109108.

Abbreviation

IID - Independent and Identically Distributed
non-IID - non Independent and Identically Distributed
RS - Recommender System
CF - Collaborative Filtering
MF - Matrix Factorization
CNN - Convolutional Neural Network
RNN - Recurrent Neural Networks
GNN - Graph Neural Networks
MNAR - Missing-Not-At-Random
MTS - Multivariate Time Series
TS - Time Series
FFN - Feed-Forward Network
GRU - Gated Recurrent Units
DFT - Discrete Fourier Transform
LSH - Locality Sensitive Hashing
DSH - Deep Supervised Hashing
CBR - Case-based Reasoning
RMSE - Root Mean Square Error
MAE - Mean Absolute Error
AP - Average Precision
MAP - Mean Average Precision
MINE - Mutual Information Neural Estimation
HT - Hit Rate
AUC - Area Under the ROC Curve

NOTATION

$\mathbf{I}$ is the identity matrix.
$\mathbb{R}$ is the set of real numbers.
$\mathcal{U}$ is the user set.
$\mathcal{I}$ is the item set.
$\mathcal{X}_U$ is the user feature set.
$\mathcal{X}_I$ is the item feature set.
$\mathbf{R}$ is the rating matrix.
$\mathbf{O}$ is the observability matrix.
$\mathbf{O}$ is the selection matrix/similarity matrix.
X is the multivariate time series input.
$\mathbf{X}$ is the input matrix.
$\mathbf{x}$ denotes a vector.
$\mathbf{X}_{i,:}$, $\mathbf{X}_i$, and $\mathbf{x}_i$ denote the ith row of matrix $\mathbf{X}$.
$x_{i,j}$ is the entry in the i-th row and j-th column of matrix $\mathbf{X}$.
$\otimes$ is the Kronecker product.
$\odot$ denotes Hadamard product, i.e., the elementwise product.
$\cdot^*$ denotes Hadamard product.
$\circ$ denotes outer product.
$\langle \cdot, \cdot \rangle$ denotes inner product.
$(.)^\top$ is the transpose operation.

TABLE OF CONTENTS

List of Figures

Figure **Page**

List of Tables

Table **Page**

Part I

Background

CHAPTER 1

INTRODUCTION

With the rapid development of information technology, a huge amount of data emerge every day in our daily life from various domains. For example, E-commerce platforms such as Taobao[1], Amazon[2], and JD[3] generate millions of transaction records per second from customers. Online social websites/applications like Twitter[4], Meta[5], and WeChat[6] connect hundreds of millions of users for social or business purposes and generate billions of posts and messages per second. Search service providers such as Google[7], Baidu[8], and Bing[9] index millions of newly-coming news, events, and terms and provide public search services for millions of users every second. Real-world big data is pervasive in our daily living, studying, working, and entertainment, and has a huge impact on the world [21].

The world's data records human behavior and embodies the laws of all trades and professions, benefiting us in better understanding the world [19]. Accumulated E-commerce records help online shopping platforms analyze customers' preferences and promote personalized shopping services [95]. Massive historical financial data attracts a large number of people to engage in quantitative investment [20]. Data has brought great

[1]https://www.taobao.com/
[2]https://www.amazon.com/
[3]https://www.jd.com/
[4]https://twitter.com
[5]https://www.meta.com/
[6]https://www.wechat.com/
[7]https://www.google.com/
[8]https://www.baidu.com/
[9]https://www.bing.com/

opportunities to the world, attracting lots of businesses to spend more than hundred of millions per year on data analysis. In such a big data era, data learning has also blossomed and dominated research communities. Plenty of global funds and efforts have been unprecedentedly devoted to data learning research and are playing major scientific roles in promoting the innovation of data learning and transforming the thinking of data science to practical applications.

1.1 Background

With the increasing attention to valuable data, more and more business organizations and research communities record, collect, sort out, and clean data from various domains with great efforts, for future business or research purposes. Data from different domains, applications, and sources are correlated to obtain broad knowledge for wide applications/research and have been sought after by data learning practitioners/researchers to deliver/study universal applications or general artificial intelligence [34, 294]. In addition, multi-modality/category/timestamp data has been generated and collected to achieve the enhanced performance of various tasks, e.g., fashion recommendation [175], autonomous driving [15], dialogue systems [280] etc., assuming there are sharing knowledge or close relationships among different modality/category/timestamp data. With the increase of data volume, data categories, modalities and dimensions also keep increasing. It is attractive but challenging for both businesses and research communities to develop comprehensive and powerful data learning methodologies and paradigms to meet to the complexity of modern real-world data.

However, the independent and identically distributed (i.e., IID for abbreviation) assumption has laid the foundation of data learning technology and engineering for several decades. IIDness simplifies real-world data's intricate nature for effectively achieving approximate, traceability, and asymptotic problem-solving [21]. An illustrative example is that a k-nearest neighborhood (k-NN) algorithm generally assumes samples are independent and identically distributed per the IID assumption and thus generates an IID learning objective and a corresponding IID learning system. Hereby, the comprehensive aspects and properties of non-IIDness, i.e., heterogeneities and couplings, of real-world samples may be over-simplified, -normalized, or -abstracted [20].

The debate on the limitations of IID assumptions has lasted for over half a century. People are struggling with whether to build IID models with high generalization and easy solutions or to specify non-IID models with high performance but complex modelings

and solutions. The demand for new paradigms of data learning has increased and expanded from statistics to informatics, computing, and other disciplines [22]. With the rapid development of data learning technology and methodology, especially deep learning techniques, more powerful models (e.g., neural networks) are devised to cater for more complicated data and application scenarios. More and more businesses and researchers focus on the complexity of real-world data and approach its non-IIDness nature, including data heterogeneity and couplings [22]. This motivates us to rethink the nature of real-world big data to satisfy its increasing amount and varying complexity.

However, non-IIDness shows diversified properties with different data scenarios, for example, heterogeneity in types, attributes, sources, and couplings within and between structures, distributions, and variables. It is extremely challenging to specify non-IID data learning methodologies for various scenarios and applications, and it is also far from reaching a unified non-IID learning paradigm for addressing various real-world non-IIDness, i.e., heterogeneity and coupling. To this end, I attempt to explore non-IID learning approaches in different specific tasks, i.e., recommender systems, multivariate time series forecasting, and learning to hash, to enlighten non-IID methodologies and techniques in this thesis. The elaborately chosen applications and scenarios penetrate our daily life living, studying, working, and entertaining, and cover various tasks of classification, ranking, representation, and retrieval. In addition, the three applications involve various non-IIDness in terms of different granularity, e.g., feature-level and label-level non-IIDness in recommender systems, variable-level and temporal non-IIDness in MTS forecasting, and instance-level non-IIDness in learning to hash. Via investigating the three applications, we accordingly deliver a comprehensive study on the non-IIDness modeling. Next, we briefly introduce the three tasks.

1.1.1 Recommender Systems

Recommender systems play increasingly important roles in various domains in the age of information explosion, e.g., e-commerce and social media, by suggesting products and services (called items in general) that match people's interests. In general, recommendations are predicted based on the analysis of existing user actions on items such as user clicks and purchases of items and user ratings, which is called *user-item interactions* and reflects user preferences for items. The user-item interactions can be characterized in terms of observable (explicit) user/item features, contextual factors and their couplings, and user sequential actions on items over time [1, 139, 276]. One major challenge is to precisely estimate missing interactions (e.g., ratings) where a large proportion of ratings

were missing [137, 233, 279] and it leads the heterogeneous interaction distributions for users and items. In addition, during the interactions, users not only maintain their stationary preferences within a certain context (e.g., in browsing a movie website, artists may prefer musical movies while children may be more excited about newly released animation) but also adapt their preferences to new or other items over time or contextual change (such as the successive release of new movies or circumstances change). It results in another important and prevalent task, e.g., the sequential recommendation, which takes a sequence of user-item interactions and tries to predict the subsequent users' actions that may happen in the near future.

1.1.2 Multivariate Time Series Forecasting

Time series (TS) forecasting has been used for many diverse real-world applications, such as economics and finance [20, 208], weather forecasting [216, 286], epidemic spread analysis [11, 186], and energy consumption planning [148, 176]. Going beyond univariate TS forecasting [113, 198], multivariate time series (MTS) forecasting characterizes the trend of multiple TS that interact, couple, and influence each other over time [20, 24]. A typical example is a cross-market analysis [24, 253], where multiple financial indicators such as the S&P 500 index, the USD exchange rate, and the FTSE 100 index couple and co-evolve. Recently, jointly modeling susceptible, infectious, and recovered COVID-19 cases coupled with external factors has emerged as a new challenge [23]. Such real-life MTS is challenging due to their complex data characteristics across MTS over variable, time, and frequency spaces. In particular, MTS is non-IID. They involve complex non-IIDnesses, including explicit-implicit, time-frequency, and short-long intra- and inter-TS coupling relationships and inter-TS heterogeneities of variable scales and distributions [3, 21, 139, 293]. Increasingly prevailing forecasting methods model intra- and inter-TS correlations simultaneously, including MTS regressions, Markovian models, the copula method, and deep neural networks (DNNs) [86, 181, 252, 253, 271]. However, such methods often universally neutralize the complexities of MTS data with partial-to-no mechanisms capturing the MTS non-IIDness. They may result in an incomplete to incorrect understanding of the nature and characteristics of MTS.

1.1.3 Learning to Hash

Due to the explosive growth of high-dimensional and large-scale data in real-world scenarios, hashing has attracted significant attention and has been widely utilized for

fast information search and retrieval tasks in recent years. Intending to improve storage and search efficiency, hashing encodes high-dimensional data into compact binary codes which preserve the similarities of original data. Parallel to traditional data-independent hashing, e.g., locality sensitive hashing (LSH) [44] applying random projections as hash functions, this thesis focuses on data-dependent hashing (specifically supervised hashing). It aims to learn task-specific hash functions to guarantee the similarities in hash coding space as close as those in the original space and is roughly categorized into supervised and unsupervised hashing [221, 222].

1.2 Challenges and Gaps

The complexity of real-world data extremely challenges the existing data learning methodologies and techniques and may seriously constrain their applicability and feasibility in practical scenarios. It is necessary and promising to consider the non-IIDness of real-world data and devise non-IID modeling and applications. However, different real-world application scenarios often bring new challenges and difficulties in non-IID modeling. This section will discuss the corresponding challenges and gaps in the non-IID modeling (i.e., coupling modeling and heterogeneity modeling) of specific applications.

1.2.1 Non-IID Recommender Systems

Complex real-world recommendation data brings great challenges and opportunities for RS communities. There are several typical challenges necessarily addressed to improve recommendation performance.

- **Heterogeneity**: In recommendation, items and users are usually associated with different features, which cannot be modeled with the same feature specification or distribution. Due to their feature differences, items and users show different item attractions and user behaviors, resulting in heterogeneous user-item interactions, e.g., explicit ratings and implicit clicks. Most current RSs are modeled under an identical distribution assumption over user/item features or user-item interactions. This often leads to poor personalized recommendation performance due to underrepresented user/item embeddings and user-item interactions. In addition, heterogeneous user-item interactions result in serious data biases, e.g., users are more likely to select and rate preferred items instead of disliked items. Recommendation models trained on biased data are more inclined to generate biased

recommendation performance, for example, generating relatively low/high rating scores or suggesting repeated/similar items. *Hence, one critical challenge in non-IID RSs is how to represent heterogeneous user|item|context features to enhance recommendation performance and model the heterogeneous user-item interactions to avoid recommendation biases.*

- **Couplings**: To construct more advanced RSs, multiple types of information from multiple data aspects have recently been incorporated to obtain more comprehensive knowledge. Increasing prevalent recommendation models focus on user-item interactions (e.g., factorization-based models), user-user interactions (e.g., social recommendation), and item-item interactions (e.g, sequential recommendation). However, those models seriously suffer from sparsity and cold-start issues. Recently, it has been proven effective to consider feature-level coupling relationships when modeling the aforementioned interactions. It brings new challenges on how to model the feature couplings among users and items where the features may be heterogeneous. In addition, it is also crucial to incorporate contextual information into the recommendation process under certain circumstances. This implies precisely modeling the coupling relationships between user features, item features, and contextual information, which has rarely been explored in previous recommendation models. *Hence, another critical challenge in non-IID recommender systems is how to well model and capture user|item|context feature couplings to enhance recommendation performance.*

1.2.2 Non-IID Multivariate Time Series Forecasting

While multivariate time series (MTS) has been studied intensively in classic TS modeling, the recent prevailing foci are on modeling MTS with intra- and inter-TS correlations simultaneously, typically by deep sequential neural networks. Real-life MTS are non-IID, which brings challenges to current MTS forecasting models due to their complex data characteristics across MTS over variable, time, and frequency spaces. They often involve explicit-implicit, time-frequency, and short-long intra- and inter-TS coupling relationships (e.g., observational correlations and hidden dependencies) and inter-TS heterogeneities (such as heterogeneous variable scales and distributions). Such non-IIDnesses with couplings and heterogeneities cannot be universally neutralized by existing methods. This motivates us to carefully devise novel MTS models to address intra- and inter-TS couplings and handle the heterogeneous time series variables. *Hence,*

the main challenge is how to address variable heterogeneity and model intra- and inter-TS couplings to enhance MTS forecasting.

In addition, recent models, e.g., matrix factorization models and GNN-based models, can attend to both the intra- and inter-TS dependencies while sequential models mainly handle the intra-series dependencies. Compared with GNN-based models, matrix factorization methods can not model the complex dependencies among time series [119]. However, GNN-based models exploit the relationships through point-wise and pair-wise interactions which can not fully express the complex relationships among time series. *It is challenging but necessary to revisit the relationships among time series (including intra- and inter-TS dependencies) from the perspective of mutual information and capture high-order and high-level coupling relationships.*

1.2.3 Non-IID Learning to Hash

To learn hash function on real-world data, there are two challenging issues necessary to consider to guarantee high-quality hash codes.

- **Code Balance**: code balance, including bit balance and bit uncorrelation, is essential to guarantee high-quality hash codes and avoid learning collapse in code generations. However, conventional code balance constraints imposed on avoiding overfitting and improving hash code quality are unsuitable for deep supervised hashing owing to their inefficiency and impracticality of simultaneously learning deep data representations and hash functions. In addition, previous data-dependent hashing models generally build objectives based on preserving the similarity relationships among the hash codes. *Few studies focus on the relationship between the original inputs and hash codes, which is necessary to improve the informativeness of hash codes.*

- **Similarity metric**: It is crucial to consider the heterogeneity embodied in complex data in similarity measurement. In heterogeneous data, attributes may follow different distributions and show different significance. Intuitively, this may lead to the inaccuracy of most handcrafted similarity measures, which adopt consistent attribute similarity or linear (e.g., average) aggregation functions [154]. To tackle the heterogeneity, previous studies optimize similarity measures by linear or non-linear aggregation functions in a data-independent manner, which cannot depict the complex attribute coupling relationships and heterogeneity among attributes [139].

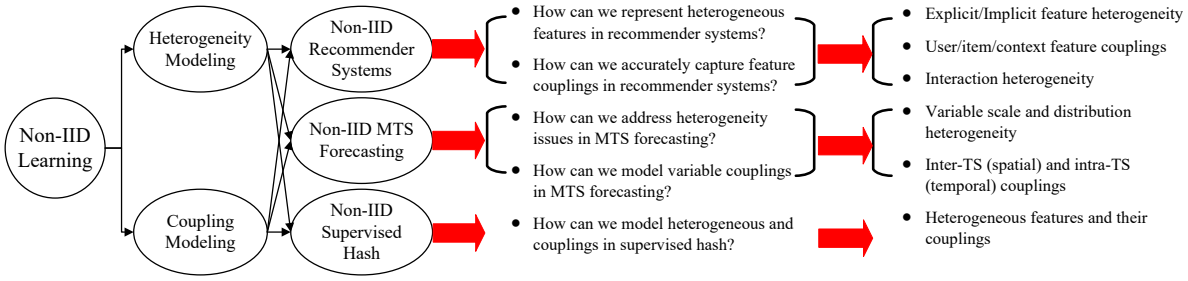


Figure 1.1: Thesis objectives of the non-IID modeling and learning approaches.

Hence, another critical challenge is how to model heterogeneous features and their couplings in learning accurate similarity metric.

1.3 Research Objectives

This thesis is mainly devoted to non-IID data learning specified to real-world applications, including studying how to deal with the aforementioned critical problems and challenges in modeling real-world non-IIDness and how to design/build specific non-IID algorithms/models to copy with non-IID applications. This thesis not only promotes the non-IID thinking of real-world complex data but also inspires data learning researchers and practitioners with advanced non-IID modeling and learning techniques in various practical applications.

As shown Figure 1.1, this thesis focuses on studying the non-IIDness from multiple aspects of modeling and applications, including coupling modeling and heterogeneity modeling, and non-IID approaches for real-world applications, i.e., recommender systems, multivariate time series forecasting, and learning to hash.

1.3.1 Coupling Modeling

Couplings refer to the relationships or interactions of two or more aspects between inputs or between inputs and outputs [22]. Modeling couplings is an important approach to capturing complex relationships in different granularity and has been empirically proven to be effective to improve model capacity. In the thesis, we aim to model explicit/implicit and inter/intra couplings and leverage coupling modeling to enhance data learning.

- Explicit/Implicit couplings: Explicit couplings refer to the relationships calculated by the surface values via various distance/similarity metrics, while implicit cou-

plings refer to the relationships calculated in the latent space or the indirect relationships obtained based on mediate variables. Both explicit and implicit couplings play important roles in data learning and reflect the direct and latent relationships respectively. It is challenging but significant to the data learning community to combine explicit coupling and implicit coupling modeling in a single model. This also has the potential to improve the model performance and model interpretability.

- Inter-/Intra- couplings: In addition to explicit/implicit couplings, inter/intra-couplings are also significant and necessary to explore complex relationships embodied in real-world data. Inter-couplings focus on the relationships of different aspects, e.g., variables and samples, while intra-couplings focus on the correlations of different internal aspects, e.g., variables or features in each sample. Inter- and intra-couplings complement each other and depict relationships from different granularity. The complementarity benefits of obtaining comprehensive relation information to enhance model performance.

1.3.2 Heterogeneity Modeling

Coupling modeling is a critical step to depict the explicit/implicit, inter/intra relationships and capture comprehensive correlations to enhance data learning. In addition, heterogeneity is built into various aspects to capture data characteristics and complexities: attribute level, variable level, and sample level. The following list explores research targets of different levels of heterogeneity in different data learning scenarios.

- Attribute level: different attributes have different categories, different scales, and different distributions. Simply treating all attributes as identically distributed easily causes failures in capturing the meaning of the attributes and modeling the fusion of the attributes. It is necessary to learn to model or embed heterogeneous attributes in a unified space or manifold allowing for attribute distance/similarity computations.

- Variable Level: generally, variables vary with each other due to their various attributes. In addition, a univariable may have different values varying over time. Treating different variables or non-static variables to follow identical or static distributions fails to capture the non-stationary manner of the variables.

- Interaction level: In addition to attribute heterogeneity and variable heterogeneity, different interactions, e.g., user's ratings and clicks on preferred or retrieved items, have large distribution differences due to the differences in the data collection. Addressing the distribution differences of interactions is necessary to learn/approximate accurate data distributions and is even beneficial to developing trustworthy data learning models.

1.3.3 Non-IID Approaches

To verify the effectiveness of our coupling modeling and heterogeneity modeling, we tailor non-IID models for various real-world applications based on the coupling and heterogeneity modelings and investigate how they can improve the performance of the model. Three representative applications are nominated for evaluation: recommender systems, multivariate time series forecasting, and learning to hash.

- Recommender systems: Recommendation is one of the most widely used data learning applications. Currently, recommender systems (RSs) seriously suffer from serious cold start, data sparsity, and data biases. Non-IID recommendation aiming to modeling the non-IID recommendation data is a potential and promising methodology and will be the key to the next-generation recommendation. To this end, we investigate and verify the effectiveness of non-IIDness modeling on recommendation performance, including exploring user-user couplings, user-item couplings and item-item couplings, and rating generation heterogeneity in *Part II Non-IID Recommender Systems*.

- Multivariate time series forecasting: Real-life MTS are non-IID, where they often involve explicit-implicit, time-frequency, and short-long intra- and inter-TS coupling relationships (e.g., observational correlations and hidden dependencies) and inter-TS heterogeneities (such as heterogeneous variable scales and distributions). It is necessary and recently prevailing to model MTS by capturing intra- and inter-TS correlations simultaneously. We aim to effectively capture both target-relevant intra- and inter-TS couplings and inter-TS heterogeneities in the variable, time, and frequency spaces to improve MTS forecasting performance in *Part III Non-IID Multivariate Time Series Forecasting*.

- Learning to hash: we aim to learn accurate hash functions to preserve the original similarity measures in hash codes. Considering the beneficial relationships between

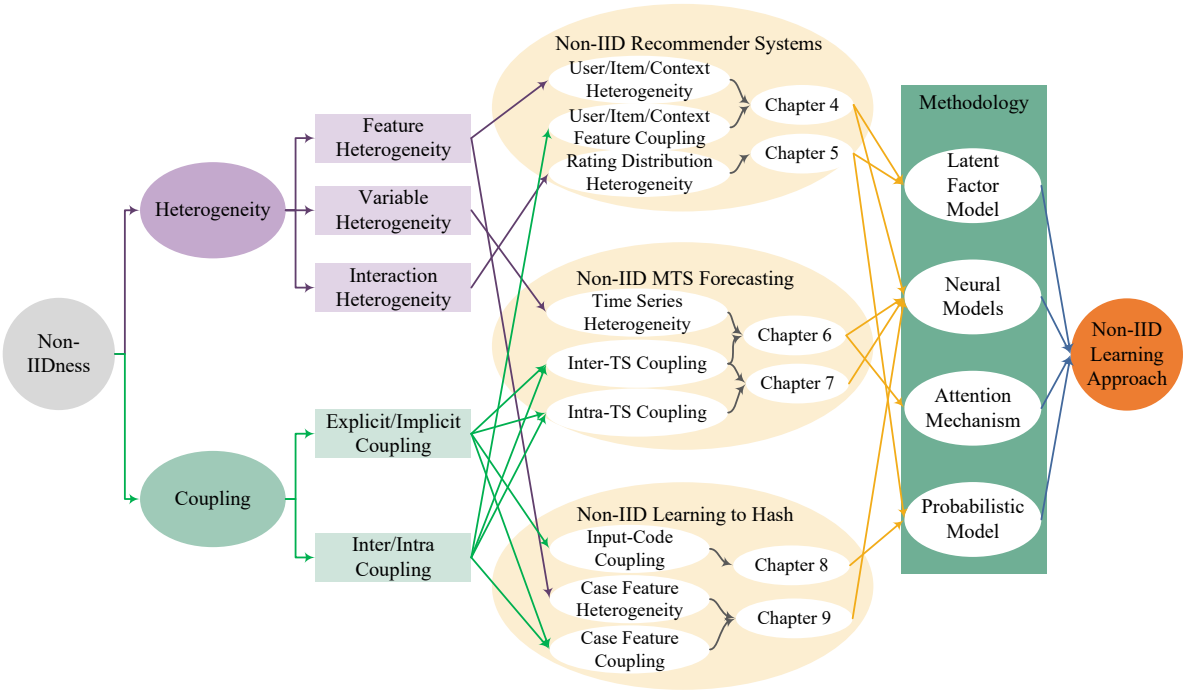


Figure 1.2: Thesis overview of the non-IID learning approaches.

the original inputs and hash codes, it is necessary to leverage the relationships to enhance the quality of hash codes. In addition, real-world high-dimensional and heterogeneous data brings great challenges and computational costs to learn an accurate hash function. We not only need to address the high-dimensionality and heterogeneity issues and also pay attention to complex inter-/intra data couplings to devise comprehensive similarity measurements in *Part IV Non-IID Learning to Hash*.

In summary, we focus on real-world non-IID data and aim to explore novel and comprehensive non-IID learning approaches for practical applications. To achieve this goal, we emphasize the necessity of modeling data heterogeneity and coupling relationships and deliver specific non-IID learning approaches for three practical tasks, i.e., recommender systems, multivariate time series forecasting, and learning to hash, respectively, as outlined in Figure 1.2. Each task has two chapters corresponding to enlighten non-IID modeling and learning approaches in which we introduce various existing methodologies, including neural network, probabilistic model, latent factor model, and attention mechanism, to achieve non-IID learning.

1.4 Thesis Organization

In this section, we briefly introduce the five parts of this thesis:

- *Part I*: This part presents the background, literature survey, and preliminaries of this thesis, and includes the following three chapters.
  - *Chapter 1*: This chapter introduces the research background, challenges and gaps, and research objectives.
  - *Chapter 2*: This chapter presents a literature review of the existing models of non-IIDness and the non-IID application scenarios and tasks. Those are most relevant to the studies in this thesis to elucidate the efficient non-IID data learning
  - *Chapter 3*: This chapter introduces the preliminary knowledge and relevant data learning techniques theories, including latent factor models, deep learning models, missing theory, discrete Fourier transform, supervised hashing, and evaluation metrics.
- *Part II*: This part studies the non-IID recommendation data and presents two non-IID learning models for recommender systems:
  - *Chapter 4*: This chapter provides a novel sequential recommendation model incorporating user preference dynamics and feature couplings. A neural time-aware recommendation network is designed based on the convolutional networks and attention mechanism to jointly learn user/item/context feature couplings and temporal heterogeneous preference, i.e., preference dynamics.
  - *Chapter 5*: This chapter provides a novel tripartite collaborative filtering model to debias rating estimate on the missing-not-at-random data. Specifically, we model the correlations among observability, selection, and rating during users' rating process and thereby build a probabilistic graphic model to eliminate the selection bias in rating estimates.
- *Part III*: This part studies the non-IIDness in multivariate time series analysis and delivers a novel cluster-enhanced neural model and a deep coupling network for non-IID multivariate time series forecasting:
  - *Chapter 6*: This chapter builds a cluster-enhanced Transformer model for non-IID multivariate time series analysis. We introduce spectral-clustering to

segregate multivariate time series and multi-channel Transformers to capture explicit-implicit, time-frequency, and short-long intra- and inter-TS coupling relationships and inter-TS heterogeneities.

  - *Chapter 6*: This chapter revisits the relationships from the perspective of mutual information and accordingly builds a comprehensive coupling model for relationships between variables. Then, we propose a novel deep coupling network for MTS forecasting, named DeepCN, which consists of a coupling mechanism to explicitly explore the relationships between variables, a coupled variable representation module to encode the different variable patterns, and an inference module to make predictions by one forward step.

- *Part IV*: This part addresses the non-IIDness in learning to hash algorithms, especially deep supervised hashing, and presents a novel learning to hash algorithm to improve hash code quality and enhance practical real-world case-based retrieval and classification, respectively:

  - *Chapter 8*: This chapter provides probabilistic code balance constraints on deep supervised hashing to force each hash code to conform to a discrete uniform distribution. Theoretical analyses reveal that the proposed constraints form a general deep hashing framework for both bit balance and bit uncorrelation and maximize the mutual information between data input and their corresponding hash codes.
  - *Chapter 9*: This chapter provides a novel deep hashing network to learn similarity-preserving compact hash codes for efficient case retrieval and proposes a deep-hashing-enabled CBR model HeCBR. In the network, we propose an efficient feature representation module to tackle heterogeneity and high-dimensionality issues.

- *Part V*: We briefly summarize this thesis and provide its contributions in this part. In addition, we discuss open challenge and provide future directions.

  - *Chapter 10*: This chapter provide a brief summary of contributions in this thesis.
  - *Chapter 11*: This chapter discuss several open challenges in non-IID learning approaches and the promising future directions for improving existing non-IID learning accordingly.

CHAPTER 2

LITERATURE SURVEY

To better understand the non-IID learning data, this chapter extensively review the existing literature in terms of models and algorithms in non-IIDness modeling, i.e., heterogeneities and couplings, the corresponding non-IID application scenarios and tasks in terms of recommendation, time-series forecasting, and learning to hash models and algorithms for improving the efficiency

2.1 Non-IIDness Modeling

In this section, we briefly review the techniques for handling non-IIDness and other data characteristics issues, e.g., high-dimensionality and data missingness. All the discussed data issues are related to the thesis.

2.1.1 Coupling Modeling

Compared with dependence, correlation, and association, coupling is a richer concept and has been widely explored and exploited in various applications [293]. It includes explicit/implicit, qualitative/quantitative, descriptive/deep, and specific/comprehensive modeling to satisfy different data characteristics and application scenarios. Zhu et al. propose an unsupervised learning model for categorical representation to jointly consider heterogeneity and coupling [293]. Zhang et al. propose time-aware neural recommendation via modeling explicit feature couplings among users/items/context to capture

stationary user preference, where the authors introduce an efficient and effective interaction network capable to learn the couplings of heterogeneous features, e.g., numerical, categorical, and time features [275]. In addition to representation learning and recommendation, many researchers introduce coupling modeling into multivariate time series analysis recently. For example, Huang et al. introduce traffic couplings via multirational graph attention networks for traffic prediction [83]. Zhang et al. construct a tree to extract hierarchical and group variable couplings and then adopt a cross attention network to jointly capture inter-series and intra-series couplings [72]. Although extensive studies verify the benefits of modeling coupling, we necessarily consider novel benchmark learning paradigms for coupling modeling, which benefits for the data analysis community to model data couplings and deliver powerful models for different complex scenarios.

In addition, feature couplings have been the key to the success of many prediction models [227] and are common in many domains, such as Click-Through-Rate (CTR), genetics studies, and environmental effects [124]. In addition to the linear effects, high-order feature couplings are also important for many complex applications [124]. Although deep neural networks (DNNs) can learn both low- and high-order feature couplings, it learns implicitly and at bit-wise level [121, 288]. There are three main categories for feature couplings, including aggregation based method, graph based method, and combination based method [288]. Compared with the other two methods, the combination method generates feature couplings explicitly. Wide&Deep [35] uses a wide component to generate cross features and takes them as input of deep neural network. DCN [227] leverages a cross network to encode feature couplings explicitly and a neural network to encode implicitly. xDeepFM [121] uses a Compressed Interaction Network (CIN) to generate feature couplings explicitly. In this section, motivated by the combination method, we model the relationships among time series explicitly via a Cartesian Product model [288].

2.1.2 Heterogeneity Modeling

Another common problem is heterogeneous data, with the underlying generation process changing across data sets or domains [273]. Wang et al. proposes a flexible information-based framework specializing the maximum entropy principle and the least effort principle to a principled multimodality information fusion formalism [218]. Liu et al. introduces transfer learning techniques to the heterogeneity between source and target domains and propose an evidence-based heterogeneous transfer classification model [138]. Zhu et al. proposes the HELIC model to capture both value-to-attribute and attribute-to-class

hierarchical couplings to reveal the intrinsic heterogeneity in data. In addition, a growing amount of research pays attention to heterogeneous information network where both attributes (or datapoints) and their relation may be heterogeneous [79, 82, 293]. Those methods model data heterogeneity and couplings and achieve significant performance improvement, proving the rationality and practicality of capturing heterogeneity.

Generally, real file high-dimensional data displays heterogeneity due to either heteroscedastic variance or other forms of non-location-scale covariate effect [223], indicating the necessity of addressing the issues of heterogeneity and high dimensionality simultaneously. Hao et al. combine the high-dimensional version of Expectation Conditional Maximization algorithm and graphical lasso penalty to jointly estimate multiple graphical models on heterogeneous and high-dimensional observations [70]. Lan et al. introduces quantile regression to model heterogeneous data and regularize quantile regression with a non-convex penalty function to deal with ultra-high dimension. Pang et al. [160] introduce a heterogeneous univariate outlier ensemble framework which ensembles a set of heterogeneous univariate outlier detectors optimized to capture different distribution of each individual feature. Inspired by the success of these methods, we tailor the proposed network to flexibly capture complex data characteristics and improve the practicality of CBR systems in real-life applications.

2.1.3 High-Dimensionality Modeling

The most common technique for handling the high-dimensionality is dimensionality reduction [53, 159], which roughly includes manifold learning [204, 299], feature selection/extraction [209, 302], and the encoder-decoder framework [106, 110, 162]. Those methods often project high-dimensional data into low-dimensional representation and expect that the representation preserves certain patterns, e.g., neighbors, distances or clusters, or maintains the maximum mutual information with the original data. Regarding the incidental sparsity issues, some recent studies [107, 245] further attempt to tackle the underfitting incurred by data sparsity. In addition, He et al. borrows the idea of factorization machine and integrates it with neural networks to handle sparsity [249]. Inspired by them, we extend the idea of factorization machine and build a multilinear interaction layer to handle the high dimensionality and sparsity. Note that a vast literature focusing on high-dimensionality is not mentioned here, please refer to [53, 159] for more details.

2.1.4 Missingness Modeling

As this subsequent explores the impact of item observability and user selection on the rating formation and the bias in estimating missing ratings in recommendation [185], below we review the related work on modeling item observability and user selection in recommendation area.

Recently, some researchers believe missing ratings reflect both non-preferred (negative) missing ratings and unobserved missing ratings [122]. They introduce a *user exposure* variable indicating whether an item was exposed to a user to joint probabilistic models and infer the exposure from user selection by the iterative estimation of user selection and the exposure [122, 136, 224, 225]. These methods distribute a confidence of being truly negative to each missing entry and then down-weight the unobserved items to avoid simply treating them as negative that are accordingly not recommended. These methods outperform the state-of-the-art CF methods for the missing-not-at-random (MNAR) data, but they only model the dependency between user selection and item observability and are tailored for recommendation with implicit feedback.

Existing models dealing with the MNAR data follow the theory of missing data in [126], which introduces a parametric joint probability distribution on the ratings and selection indicator. For example, CPT-v and Logitvd [146, 147] use a Mixture of Multinomials (MM) to generate user rating values and model user selection based on these values. More recently, RAPMF [125], MF-MNAR [77] and SPMF-MNAR [29] leverage the powerful probabilistic matrix factorization (PMF) to model user ratings and selection, and SPMF-MNAR further applies social influence rather than just the rating to generate user selection. However, these models neglect the influence of item observability on user selection and treat that all missing entries equally as unselected. This treatment may introduce bias as the missing values actually contain both non-preferred and unobserved entries. Furthermore, these models only consider the dependency of user selection on rating values but fail to reveal the intrinsic multifaceted correlation embodied between user ratings and selection.

In addition, some methods address the MNAR problem by computing an estimated error of the prediction error of imputed values on missing entries [200, 213, 233]. These methods often have a large bias due to imputation inaccuracy, which is then propagated into training and degrade the performance. Some other recent methods [94, 177, 185, 202, 260] leverage causal inference to handle the MNAR problem. These methods leverage the inverse propensity score (IPS) for each observed entry to propose an unbiased estimator for model training and evaluation. They are suitable for recommendation of either explicit

or implicit feedback and have been theoretically and are empirically demonstrated effective and robust. However, IPS-based methods, different from our method and the aforementioned missing theory-based methods, often suffer from the high variance of the propensities [207] and extra metadata may be necessary for estimating the propensity. To the best of our knowledge, no deep learning models are available for MNAR rating estimation, thus deep models are not considered in the experiments despite of these outstanding performance in rating estimation.

2.2 Recommender Systems

In this section, we briefly review related recommendation models, including collaborative filtering, neural recommendation models and sequential recommendation models.

2.2.1 Collaborative Filtering

Matrix factorization (MF) [265] models, in particular, have obtained dominance in the recommendation community and have shown their superiority by winning the Netflix Prize competition. Basically, MF methods factorize the user-item rating matrix using low-rank approximations and use user/item latent factors for prediction. So far, various matrix factorization methods have been proposed, for example probabilistic matrix factorization (PMF) [179] and maximum-margin matrix factorization [199]. Apart from the MF approach, other models have also achieved success in recommendation. With the prevalence of Deep Learning techniques, Restricted Boltzmann Machines have been successfully adopted in collaborative filtering [60] and have achieved desirable performance to in the Netflix Prize competition [180]. Recently, various neural networks have been adopted to enhance the capacity collaborative filtering, such as neural collaborative filtering [76], deep matrix factorization [254], neural graph collaborative filtering [232] etc. Those neural collaborative filtering models explore the integration of neural networks with collaborative filtering and achieve state-of-the-art recommendation performance.

2.2.2 Neural Recommendation Models

Due to the powerful capability of data fitting of deep neural networks, current state-of-the-art recommendation approaches are mainly based on neural networks, such as Recurrent Neural Networks (RNN) [240], Convolutional Neural Networks (CNN) [205], Graph Neural Networks [246], memory networks [32], attention networks [300] etc.

These deep neural networks have been applied into various recommendation tasks, such group recommendation, session-based recommendation, sequential recommendation, and cross-domain recommendation. Next, we introduce the related work on CNN and attention networks which are close relevant to the modeling in the thesis.

Different from RNN, Convolutional neural networks (CNN) first put all the embeddings of these interactions into a matrix, and then treats such a matrix as an ‚Äúimage‚Äù in the time and latent spaces. Therefore, the CNNs can learn sequential patterns as local features of the image using convolutional filters for the subsequent recommendations. CNN-based models such as Caser [205], NextItNet [267], and [251] successfully design a CNN-based networks and achieve state-of-the-art performance in various recommendation tasks including next-item prediction, session prediction and next-basket prediction. The promising results are attributable to the fact that a CNN does not have strong order assumptions over the interactions in a sequence, and they learn more complex sequential patterns, such as skip patterns and union patterns.

The models most relevant to our work are Recurrent Recommendation Networks (RRN) [240], ConvolutionAl Sequence Embedding Recommendation Model (Caser) [205] and Sequential Temporal context-Aware Recommendation (STAR) [169]. RRN models both item changes and user preference changes by two separative LSTM auto-regressive models in addition to a low-rank factorization model upon user/item variables to capture stationary preferences. Caser captures sequential patterns by a convolutional network and combines a user latent vector to model user stationary preferences. Both RRN and Caser do not consider the influence of temporal context on user preference dynamics. SITAR based on STAR involves (temporal) context in stacked RNNs for sequential recommendation and captures the dynamics of contexts and temporal gaps. Furthermore, these three models neglect the interactions between explicit users, items and temporal context and suffer from modeling user stationary preferences and alleviating cold-start issues. Different from the above models, our proposed TARN models user stationary preferences by capturing the interactions between explicit user/item/temporal features and involving temporal context into convolutional networks to model user preference dynamics, which improves model comprehensibility and recommendation performance. Considering that CNNs are used to capture multi-fold sequential patterns in TARN, we also investigate the recent CNN-based methods for fashion recommendation [30, 95, 175, 231], hashtag recommendation [62], and news and document recommendation [101, 206]. These methods apply CNNs to process images and text, which is quite different from our work utilizing CNNs to model user action sequences.

2.2.3 Sequential Recommendation Models

With extra user/item/context features becoming available, it has been increasingly recognized that involving such features is critical for understanding the nature and challenges of recommendation. The user/item/context features and their coupling relationships [276, 292, 293] has been increasingly considered in recommendation research since they embodied user/item appearances and context influences, which is contributive to the challenges of recommendation [1]. Previous work such as factorization machines (FM) [215] captures latent factors in a low-dimensional space and factorize both first- and second-order feature couplings via the inner product of feature embedding vectors. Recent work such as NFM [75], DeepFM [67] and DCN [227] further feeds latent feature embeddings into deep networks to capture higher-order and nonlinear feature couplings. Such neural models capture latent relations between ratings and latent features and outperform the aforementioned shallow recommenders. Due to their effectiveness and efficiency in highly sparse case, these models are successfully adopted for recommendation and prediction. However, these shallow and neural models only learn stationary user preferences but ignore user preference dynamics over time, thus they are impractical for real-life applications.

There have been an increasing number of models incorporating temporal dynamics into user preference modeling, which can be roughly categorized into two basic approaches: temporal modeling [175, 203, 263] and sequential modeling [96, 205, 219, 240, 251] which is also our focus in this work. Typical sequential modeling methods include Markov chain-based methods such as [74, 172] which factorize the transition matrix of user successive actions to capture the transitional patterns of user action, which are not suitable for capturing high-order sequential patterns and the long-term dependency between user actions. The recent neural network-based models such as RNN-based [84, 219, 240] and CNN-based [205, 251] models prevail over the early sequential models and achieve excellent results in sequential recommendation. Their success relies on the strong representation capability of deep neural networks and their sequential modeling through capturing precise action transition patterns to properly represent user preferences and dynamics. With the great success of the attention mechanism in modeling global dependencies between input and output [116, 210, 300], attention neural networks have been introduced into sequential recommendation and achieve state-of-the-art performance and good interpretability.

However, the above models either ignore user stationary preferences or only model stationary preferences by factorizing latent user-item interactions without utilizing

explicit user/item/context features. Models which do not capture feature couplings cannot characterize the intrinsic cause of interactions between users and items and may greatly suffer from cold-start problems. In addition, recent advanced deep neural models such as graph neural networks (GNNs) [168, 244], memory networks [32, 291] and deep reinforcement learning [226] have been introduced into SRSs and achieve great success due to their strength for modeling and capturing the comprehensive relations within user action sequences [230]. As these models take different mechanisms and designs from our proposed TARN, we will not compare them with TARN in the experiments.

2.3 Time Series Analysis

Existing methods for time series analysis can be grouped into two categories: *univariate methods* [158, 274] and *multivariate methods* [18, 188]. Instead of focusing on univariate methods for single TS without considering their correlations, we focus on MTS and those multivariate methods which use deep learning techniques.

2.3.1 Multivariate Time Series Forecasting

Traditional MTS models such as vector autoregression (VAR) [237], matrix factorization [266], and probabilistic models [31, 187] involve appealing interpretability and theoretical guarantees. They, however, may fall short in capturing nonlinear MTS relations and are not able to adapt to large-scale multivariate data [37, 49].

Specifically, some previous work apply matrix factorization methods to factorize the relationships between time series into a low rank matrix, and then perform the forecasting in the low-dimensional latent space. TRMF [266] incorporates temporal regularization into matrix factorization formulation. DeepGLO [188] introduces TCN as a regularization to add non-linear based on TRMF. TLAE [155] advances the global factorization approaches and offers an efficient combination between flexible nonlinear autoencoder mapping and inherent latent temporal dynamics. However, these matrix factorization methods fall short of exploiting the structural dependencies among time series [119].

Recent deep learning models have gained popularity in MTS forecasting. A long- and short-term network (LSTNet) [111] leverages CNNs to discover local dependency patterns among multiple series and LSTMs to capture their complex temporal dependencies. A multilevel construal neural network (MLCNN) [37] derived from the construal level

theory leverages CNNs to generate multilevel feature abstraction over MTS and employs LSTM for multiple predictive tasks. Spatio-temporal graph convolutional networks (ST-GCNs) [264] integrate graph convolution and gated temporal convolution through spatio-temporal convolutional blocks for traffic prediction. Spadon et al. [198] further propose a recurrent graph evolution neural network (ReGENN) to infer multiple multivariate relations between co-occurring time series.

More impressively, MTS have embraced GNN [6, 247, 248, 264] because of their best capability of modeling inter-dependencies among time series. DCRNN [120] leverages bidirectional graph random walk to capture the inter-dependencies among variables. STGCN [264] integrates graph convolution and gated temporal convolution for traffic forecasting. GraphWaveNet [248] captures spatial-temporal dependencies efficiently and effectively by combining graph convolution with dilated casual convolution. AGCRN [6] proposes a data-adaptive graph generation module and a node adaptive parameter learning module to enhance graph convolutional network. MTGNN [247] presents an effective learning method to exploit the inherent dependencies among variables.

Further, the attention mechanism aligns temporal sequences and extracts long-range dependence [72, 167]. For example, CATN [72] adopts a cross attention mechanism based on a tree structure to learn hierarchical temporal correlations. The most related and representative work is Transformer-based MTS models which apply vanilla Transformer to capture intra-TS temporal correlations [196, 271]. Recently, sparse self-attention has been used to reduce the computational cost of Transformer and tackle the challenge of long time series forecasting [104, 118, 243, 289]. Informer [289] introduces a novel ProbSparse mechanism and a distilling operation for time- and memory-efficient self-attention calculation. Autoformer [243] proposes an efficient auto-correlation mechanism with a decomposition architecture to progressively aggregate long-term trends for prediction. FEDformer [290] subsequently introduces a novel attention mechanism with low-rank approximation in frequency. The above models have achieved great success in MTS forecasting due to their stronger capability in capturing temporal relations between multiple variables. However, they neglect the heterogeneity of variables and rarely capture complex inter-TS couplings over time and frequency.

2.3.2 Time Series Clustering

Time series classification and clustering are essential tasks across many domains and applications [4, 189]. In contrast to classification relying on the supervision of labels, time series clustering learns time series relevance without supervision, which is applicable

to MTS forecasting. It segregates time series variables such that relevant variables are clustered while irrelevant variables are apart [115]. Existing algorithms for time series clustering can be generally categorized into two families: *raw-data-based* and *feature-based* methods [145, 214]. Raw-data-based methods adjust the distance function to capture sequential characteristics, which are vulnerable to noise and insensitive to temporal information [163, 259]. Feature-based methods extract feature representations of time series and learn to cluster based on extracted features, which effectively alleviates the influence of noise and outliers [145]. Feature-based methods include two categories: 1) two-stage approaches that cluster after extracting features [66, 270]. These approaches usually introduce prior knowledge to pre-processing raw data to enhance feature quality; 2) one-stage approaches that jointly learn feature representation and clustering [143, 145]. The approaches are capable of extracting nonlinear high-level features. Taking the advantages of both kinds of feature-based methods, we propose a hybrid clustering method that first utilizes DFT to obtain features in the frequency domain and then applies neural networks guided by the K-means objective to learn high-level features.

2.3.3 Spectral Analysis for Time Series Data

Spectral analysis has been widely used in TS analysis. It has advantages in decomposing TS into multiple independent variables and capturing periodic patterns within sequential data [105, 113]. Recently, spectral analysis has been incorporated into neural networks for time series analysis. StemGNN [18] first incorporates DFT with graph neural networks and models inter-TS correlations and temporal patterns jointly in the spectral domain. STCN [144] employs spectral analysis to constrain similar features to obtain the same pseudo-class labels. Autoformer [243] and FEDformer [290] utilize a fast Fourier transform to calculate the correlations between TS. In this work, we use a Fourier transform to extract spectral features and then correlate and cluster MTS based on the extracted features to enhance forecasting performance.

2.4 Learning to Hash

In this section, we briefly introduce related work on learning to hash (i.e., data-dependent hashing) and then review recent models for traditional supervised hashing and deep supervised hashing.

2.4.1 Data-dependent Hashing

Parallel to the data-independent hash, this thesis mainly discusses data-dependent hashing from the perspective of non-deep hashing and deep hashing.

Data-dependent hashing achieves superior performance and learns the similarity-preserving hash functions from data by minimizing the gap between the similarity in the original space and that in the hash code space. Early studies present various approaches focusing on non-deep hashing, which relies on handcraft features to learn hash codes and hash functions. According to hash functions, those methods are roughly categorized into linear hash functions, kernel hash functions, and eigenfunction hash functions [222]. For example, spectral hashing [238] and hashing with graphs [135] are representative algorithms with eigenfunction hash functions. ICA hashing [73] and LDA hashing [197] are linear hashing methods. Non-deep hashing often introduces code balance constraints to avoid learning collapse and facilitate the generation of compact hash codes, achieving desirable performance in similarity retrieval.

With rapid progress made in deep representation learning, deep hashing has achieved significantly better performance than non-deep hashing and has thus been widely applied. Deep hashing methods build neural hash functions to obtain robust and powerful feature representations for complex data and learn neural hash functions and hash codes simultaneously [130]. The earliest work in deep hashing, semantic hashing [178], utilizes a deep generative model to handle text data. Subsequently, extensive studies, e.g., [26, 257, 297], introduce successive CNN-based networks to capture high-level features from images and customize learning objectives to preserve similarity. Inspired by these deep hashing methods, we build a specific neural network to learn feature representations from high-dimensional and heterogeneous data and utilize a pairwise loss function to learn neural hash functions and hash codes.

Data-dependent hashing methods are roughly categorized into supervised and unsupervised hashing. The former as our research focus in the thesis is discussed in this section.

2.4.2 Traditional Supervised Hashing

Supervised hashing aims to learn similarity-preserving hash functions via the given/-computed similarity relation in the original space [222]. With the emergence of early supervised hashing methods, e.g., spectral hashing [238], graph hashing [133] and kernel hashing [134], which learn hash projection vectors rather than random projections in the

data-independent hash, supervised hashing has attracted an increasing amount of research interest and achieved significantly better performance than the data-independent hash [192]. To improve hash quality, the non-deep supervised hashing adopts code balance constraints – bit balance and bit uncorrelation – to avoid learning collapse and facilitate generating compact hash codes [142].

2.4.3 Deep Supervised Hashing

Recently, deep supervised hashing leveraging the capability of deep learning in representation learning, outperform non-deep hashing methods and has thus been widely applied [50, 242]. Most DSH methods focus on refining or customizing objective functions, e.g., quantization loss [58, 297], weighted pairwise loss [26] and triple loss [127], for preserving similarity. A couple of works, e.g. pointwise methods [92, 201], further introduce label supervision to capture global position relationship, which is influenced by the quality of classification results. Recently, some works introduce class centers as proxies of classes to ensure continuous semantic similarity, which improves the discriminability of hash codes [228, 268]. Although these methods can achieve excellent performance, code balance constraints, which are beneficial to improving hash quality, are generally ignored in the DSH context. Quite a few previous methods such as [33] consider code balance to improve hash quality. However, those methods often utilize discrete optimization algorithms to solve the objective functions and act on the whole data population, which thus is inefficient under large-scale data and unsuitable for DSH. To tackle these issues, this work proposes a probabilistic code balance suitable for DSH to facilitate the learning of compact and informative hash codes.

2.5 Case-based Reasoning

Case-based reasoning can be categorized into attributed-based and structure-based approaches in terms of case representation methods. Next, we will introduce related work about the two kinds of approaches.

2.5.1 Traditional Case-based Reasoning

Attributed-based approaches avoid complicated computation in building representation structures, but they also require more elaborate techniques for optimizing similarity measures. In [174], a compound distance approach measures the similarity on mixed data

in CBC. Their method computes the distances and weights for categorical and numeric features by different means and achieves reasonable interpretability and accuracy on mixed datasets without data transformation. In [13], an artificial neural network (ANN) allocates attribute weights. It first trains the neural network and then treats connection weights as the corresponding attribute weights in CBC. It has such shortcomings that an enormous amount of training data is needed and the network structure cannot be transferred to different problems.

Heuristic methods, such as simulated annealing (SA) [150] and genetic algorithm (GA) [80], are applied to optimize weight allocations. These methods utilize evolutionary control optimization of classification hits on training data to iteratively update attribute weights. Subject to initial settings, however, these two methods cannot guarantee a global optimum. To improve the rationality of weight distribution, a membrane computing-based approach, named MCCBR [255], first iteratively optimizes weight allocation according to predefined evolution and communication rules, and a regional sub-algorithm based on SA is then applied to optimize the weight allocation. Our literature review shows that all these methods barely consider CBM and retrieval efficiency issues and conduct experiments with only other CBC methods but not typical classification methods, e.g., SVM and kNN. Besides, it also shows that attribute vectors-only approaches may lead to restrictive optimization of similarity measures.

Structure-based approaches organize cases and build case indexes, which have shown effective in improving the performance and retrieval efficiency of CBR systems. For example, the CBR system Déjá Vu [194] organizes software-design cases by hierarchically storing the outlines and details of the solution in different layers. The system achieves desirable performance, but relies on experts for its construction and leads to low generability. An effective and efficient Z indexing approach [129] indexes and retrieves cases and divides the case base into small sets in a tree to narrow down the search range. A growing hierarchical self-organizing map (GHSOM) [68, 296] categorizes similar cases into the same clusters and indexes the clusters. GHSOM improves the CBR efficiency and accuracy, but the system was not compared with typical classifiers for accuracy.

The aforementioned methods neglect the case relations embodied in the representation structure. The structural information often promotes similarity measures and the performance of CBR systems [141]. However, limited research is on the advantages of the structural correlations between cases in the representation structure. Accordingly, the Least-Common Subsumer (LCS) trees [183] organize plan cases and define refinement operators to compute similarity based on the structural information in LCS trees.

It achieves good accuracy and efficiency in plan retrieval and shows the potential of structural information to improve the similarity measure.

2.5.2 Scalable Case-based Reasoning

Our work aims to improve the efficiency and scalability of CBR systems with high-dimensional and heterogeneous data by leveraging supervised deep hashing. We therefore first discuss the state-of-the-art methods aiming to improve the efficiency and scalability of CBR systems. Then, we provide a brief review of the current studies on handling the high-dimensionality and heterogeneity-related issues. In addition, we also present the recent progress on the data-dependent hashing.

Due to the increase of case number in case base during retaining the newly solved cases, the efficiency and scalability of CBR systems have always been crucial hindering the CBR systems from being exploited in applying large-scale and complex cases. Early studies pay attention to the strategies of case retention which expect to reduce redundant cases and maintain informative cases only [46, 193, 277]. Those methods leverage similarity-threshold filters to prune redundant cases or perform structure reduction to refine the structure of the case base, which alleviates the rapid growth of the case base to some extent. However, they highly rely on domain experience or expertise to calculate the similarity and set threshold and degrade the applicability of CBR systems on large-scale real life data.

A more promising approach to handle the efficiency and scalability issues is to build well-organized structures to index cases. For example, the typical CBR system [195] organizes software-design cases by hierarchically storing case description and the solutions in different layers and achieves desirable performance and efficiency. Liu et al. [128] introduce an effective and efficient Z indexing approach to index cases and divide the case base into small sets in a tree. Ruiz et al. [182] propose Least-Common Subsumer (LCS) trees to organize plan cases. Those methods leverage a hierarchical tree structure to index cases and narrow down the search space for the goal of efficiency improvement. In addition, many researchers adopt clustering techniques to accelerate case retrieval. A growing hierarchical self-organizing map (GHSMO) [69, 295] is introduced to categorize similar cases into same clusters and then index the clusters. Muangprathub et al. and Zhang et al. further introduce a complete concept lattice for conceptual clustering to structurally organize and index cases. Like the structure-based CBR models, the clustering-based models improve the efficiency and accuracy of CBR systems, however all those models cost a large amount of time to construct and update the organizational

structure and index and often require extra storage to maintain the structure and index, leading to low applicability in the real world.

Hashing techniques as a special indexing approach has also been applied into CBR systems. Hashing methods, e.g., LSH [87, 281] and E2LSH [44], project (high-dimensional) cases into low-dimensional binary vectors (hash codes) and maintain the similarity information from the original space. The methods not only provide an efficient index of cases also enable approximate nearest neighbor search in the hash space and substantial data compression for the case base. Most previous studies introduce only LSH (data-independent hashing) as the underlying hash-based nearest neighbor search algorithm for large-scale cases. For example, Jalali et al. presents a case study using Map Reduce and LSH to make the ensembles of adaptation of regression (EAR) in CBR feasible for large case bases and subsequently develops foundational scale-up methods using LSH for fast approximate nearest neighbor search of both cases and adaption rules for industrial scale prediction [88]. Woodbridge et al. introduce LSH as an alternative to improve biomedical signal search results and accelerate search speed. Those methods prove the effectiveness of hashing methods in fast similarity search in CBR systems and also benefit performance improvement. However, data-independent hashing has its intuitive drawbacks that it often needs longer hash bits and cannot capture data characteristics. Recently, Jiang et al. design a supervised hashing method based on linearly combined kernel functions associated with individual features from images for scalable histopathological image analysis in a CBR system [91] to handle image data with efficiency improvement. Nevertheless, there exist quite few studies adopting advanced data-dependent hashing techniques in CBR systems.

CHAPTER 3

PRELIMINARIES

This thesis aims to explore practical Non-IID data learning approaches, we first introduce some preliminaries of the data learning models and then present some relevant theories and techniques. All those models, theories, and techniques are relevant to the learning approaches in the following chapters. Specifically, this chapter introduces latent factor models, deep neural models, missing theory, discrete Fourier transform, supervised hashing, and evaluation metrics for different learning tasks.

3.1 Latent Factor Models

In this section, we introduce basic notions and methodology of two typical latent factor models, i.e., probabilistic matrix factorization and factorization machine.

3.1.1 Factorization Machine

Factorization Machine (FM) is proposed by Steffen Rendle [215] and is one of generic supervised learning models. It maps arbitrary real-valued features into a low-dimensional latent factor space and can be applied naturally to a wide variety of prediction tasks including regression, classification, and ranking. Intuitively, it is capable to handle complex data with heterogeneous features, e.g., both categorical and numeric features. Empirically and theoretically, FMs can estimate model parameters accurately under very sparse data and train with linear complexity, allowing them to scale to very large

datasets. The aforementioned characteristics lead FMs an ideal alternative for real-world applications, e.g., recommendation [215] and clickthrough rate prediction [75].

Specifically, given input data $\mathbf{x} \in \mathbb{R}^n$, a factorization machine of degree $d = 2$ is defined as:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{3.1}$$

where $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^n$ and $\mathbf{V} \in \mathbb{R}^{n \times k}$ are model parameters, and k is the dimension of latent factors. And $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \tag{3.2}$$

where $\mathbf{v}_i \in \mathbf{V}$ is the i-th row in $\mathbf{V}$ corresponding to the i-th variable with k factors.

3.1.2 Probabilistic Matrix Factorization

Probabilistic matrix factorization (PMF) [179] is a probabilistic model to illustrate the matrix factorization approach. Given a matrix $\mathbf{Y} \in \mathbb{R}^{N \times M}$ with N users and M items, we can assume the distributions with the d-dimensional latent factors $\mathbf{U}_i \in \mathbb{R}^d$ of users and $\mathbf{V}_j \in \mathbb{R}^d$ of items:

$$P(\mathbf{U}_i) = \mathcal{N}(\mathbf{U}_i | \mathbf{0}, \sigma_U^2 \mathbf{I}) \qquad P(\mathbf{V}_i) = \mathcal{N}(\mathbf{V}_j | \mathbf{0}, \sigma_V^2 \mathbf{I}) \tag{3.3}$$

$$P(Y_{ij} | \mathbf{U}_i, \mathbf{V}_j) = \mathcal{N}(Y_{ij} | \mathbf{U}_i^\top \mathbf{V}_j, \sigma^2) \tag{3.4}$$

$$P(\mathbf{U}, \mathbf{V} | \mathbf{Y}) \propto P(\mathbf{Y}, \mathbf{U}, \mathbf{V}) = \prod_{i,j} P(Y_{ij} | \mathbf{U}_i, \mathbf{V}_j) \prod_i P(\mathbf{U}_i) \prod_j P(\mathbf{V}_j) \tag{3.5}$$

where $\mathbf{U} = [\mathbf{U}_1, \cdots, \mathbf{U}_N]$ denotes the user factors; $\mathbf{V} = [\mathbf{V}_1, \cdots, \mathbf{V}_M]$ denotes the item factors; and $\sigma_U^2, \sigma_V^2, \sigma^2$ are the variance parameters of the Gaussian distributions. Accordingly, we have the posterior $P(\mathbf{U}, \mathbf{V} | \mathbf{Y}) \propto P(\mathbf{Y}, \mathbf{U}, \mathbf{V})$ given in Equation 3.5. Then the objective function can be obtained by minimizing the negative log-posterior:

$$J = \arg\min_{\mathbf{U}, \mathbf{V}} \frac{1}{2} \left[\sum_{i,j} (Y_{ij} - \mathbf{U}_i^\top \mathbf{V}_j)^2 + \lambda (\|\mathbf{U}_i\|_2^2 + \|\mathbf{V}_{j2}^2)\| \right] \tag{3.6}$$

where $\sigma^2 = 1$ and denote $\lambda = \sigma_U^{-2} = \sigma_V^{-2}$. To solve this objective function, we can adopt the alternating least squares (ALS). When fixing $\mathbf{U}$, it yields:

$$\mathbf{V}_j \leftarrow \left(\lambda \mathbf{I} + \sum_i \mathbf{U}_i \mathbf{U}_i^\top \right)^{-1} \sum_i Y_{ij} \mathbf{U}_i \tag{3.7}$$

Similarly, when fixing $\mathbf{V}$, we can obtain:

$$\mathbf{U}_j \leftarrow \left(\lambda \mathbf{I} + \sum_j \mathbf{V}_j \mathbf{V}_j^\top\right)^{-1} \sum_j Y_{ij} \mathbf{V}_j \tag{3.8}$$

3.2 Deep Learning Models

In this section, we briefly introduce convolutional neural networks and Transformer that are closely related to the models in the thesis.

3.2.1 Convolutional Neural Networks

In deep learning, a convolutional neural network (CNN) is a classical neural network, most commonly applied to computer vision. CNNs are also known as shift/space invariant neural networks and adopt shared weights of the convolution kernels or filters that slide along inputs and provide translation-equivalent feature maps. Considering the dimensionality of inputs, there exists 1D-, 2D, and 3D-convolutional filters in the widely-used CNNs.

Specifically, given input $\mathbf{X} \in \mathbb{R}^{n \times d}$, the j-th convolution feature by the 1D wide convolution[1] is given by:

$$\mathbf{c}_i^j = \phi(\mathbf{f}_i \cdot \mathbf{X}_{j:j+h-1} + b), \quad , s.t. \quad j \in \{1, 2, \cdots, n\} \tag{3.9}$$

where each $\mathbf{f}_i \in \mathbb{R}^{h \times d}$ is h-width and d-length filter and with l different filters $\{\mathbf{f}_1, \cdots, \mathbf{f}_l\}$. Here, operator $\cdot$ computes the inner-product, b is a bias term, and ϕ is a non-linear activation function (e.g., $ReLU$). $\mathbf{X}$ is extended to $\mathbf{X} \in \mathbb{R}^{(n+h-1) \times d}$ with 0 padding for wide convolution. Each filter f_i slides along the input $\mathbf{X}$ to produce a feature map: $\mathbf{C}_i = [\mathbf{c}_i^1, \mathbf{c}_i^2, \cdots, \mathbf{c}_i^n]$ for the i-th filter. Generally, convolutional networks may include local and/or global pooling layers along with convolutional layers. Pooling layers reduce the dimensions of feature maps and effectively activate/capture informative feature for the next layer.

3.2.2 Transformer

A transformer is a deep learning model that adopts the mechanism of self-attention. It have been used primarily in the fields of natural language processing and computer

[1] Wide convolution guarantees the same heights of convolutional input and output and retains more feature combinations near the end of the sequence of filter.

vision. The self-attention used in Transformer is the scaled dot-product attention defined on the tuple input (queries $\mathbf{Q}$, keys $\mathbf{K}$, values $\mathbf{V}$) [210], where queries and keys have d_k dimensions and values have d_v dimensions. The dot-product of queries with keys is scaled by $\frac{1}{\sqrt{d_k}}$ and normalized to obtain the weights using softmax functions. Then, the outputs of self-attention are calculated by attending to values referring to the weights, formulated as follows:

$$\mathscr{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{3.10}$$

Under the multihead settings, d_m-dimensional queries, keys and values are projected to d_k, d_k, and d_v dimensions for h times by different linear projections respectively. Multihead attentions are then performed on these tuple inputs in parallel and yield $d_v * h$ dimensional output values, generally holding $d_v = d_k = d_m/h$. Then, the multihead attention is formulated as follows: Given input $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$,

$$\begin{aligned} \text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \text{Stack}(\text{head}_1,\cdots,\text{head}_h) \\ \text{head}_j &= \mathscr{A}(\mathbf{Q}\mathbf{W}_j^Q,\mathbf{K}\mathbf{W}_j^K,\mathbf{V}\mathbf{W}_j^V) \end{aligned} \tag{3.11}$$

where $\mathbf{W}^Q \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}^K \in \mathbb{R}^{d_m \times d_k}$, and $\mathbf{W}^V \in \mathbb{R}^{d_m \times d_v}$ are projection parameters, and head_j denotes the j-th single head attention. The operator of Stack stacks the attention outputs of each head. For multi-layer attentions, the output is then fed to feed-forward networks followed by layer normalization [5] and residual connections. The calculation is formulated as:

$$\mathscr{Z}^l = \text{LayerNorm}(\mathscr{Z}^{l-1} + \text{MultiHead}(\mathscr{Z}^{l-1})) \tag{3.12}$$

where $\mathscr{Z}^l$ denotes the inputs of the l-th attention layer. Then $\mathscr{Z}^l$ is fed into the feed-forward networks with two linear transformations by a ReLU activation in between:

$$\mathscr{Z}^l = \text{FFN}(\mathscr{Z}^l) = \text{ReLU}(\mathscr{Z}^l \times_T \mathbf{W}_1^l + \mathbf{b}_1^l) \times_T \mathbf{W}_2^l + \mathbf{b}_2^l \tag{3.13}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_m \times d_{ff}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{ff} \times d_m}$.

3.3 Data Missing Theory

The main principles of the theory of missing data developed in [126] have been widely studied in the context of rating data. We first formalize the theory of missing data and then introduce its application in debiasing recommendation models.

Given rating matrix $\mathbf{R}$ where $r_{i,j} \in \mathbf{R}$ denotes the rating provided by user i on item j. In practice, we collect the rating $\mathbf{R} = \{\mathbf{R}^o, \mathbf{R}^{\neg o}\}$, where $\mathbf{R}^o$ and $\mathbf{R}^{\neg o}\}$ denote the sets of observed and missing entries in $\mathbf{R}$ respectively. For each $r_{i,j}$, there defines a Bernoulli random variable $x_{i,j}$ that indicates where $r_{i,j}$ is observed ($x_{i,j} = 1$) or not ($x_{i,j} = 1$), and $\mathbf{X}$ denotes all the $x_{i,j}$ corresponding to the $r_{i,j}$. In the missing theory, it is assumed that $\mathbf{R}$ is generated by a complete data model (CDM) with parameters Σ, and is generated by a missing data model (MDM) with parameters Ω. Both models may share a set of latent variables $\mathbf{Z}$. Given Σ and Ω, the corresponding joint distribution for $\mathbf{R}$,$\mathbf{X}$,$\mathbf{Z}$ is given:

$$p(\mathbf{R},\mathbf{X},\mathbf{Z}|\Sigma,\Omega) = p(\mathbf{X}|\mathbf{R},\mathbf{Z},\Omega)p(\mathbf{R},\mathbf{Z}|\Sigma) \tag{3.14}$$

Most machine learning focuses on the estimation of the CDM given by $p(\mathbf{R},\mathbf{Z}|\Sigma)$, which MDM formalized by $p(\mathbf{X}|\mathbf{R},\mathbf{Z},\Omega)$ is normally ignored. In mechanisms for missing data are usually divided into three classes: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). MCAR is the most restrictive assumption, where the probability of observing a rating is independent of the value of any rating or latent variable generated by the CDM, that is, $p(\mathbf{X}|\mathbf{R},\mathbf{Z},\Omega) = p(\mathbf{X}|,\Omega)$. In MAR data the observation probability depends only upon the value of the observed data and the MDM parameters, i.e., $p(\mathbf{X}|\mathbf{R},\mathbf{Z},\Omega) = p(\mathbf{X}|\mathbf{R}^o,\Omega)$. MAR is popular in machine learning where the MDM is ignored without introducing any bias. However, in the general case of MNAR data, $\mathbf{X}$ is not independent of $\mathbf{R}$ or $\mathbf{Z}$ where MDM model cannot be ignored since the binary matrix $\mathbf{X}$ has dependence with $\mathbf{R}^o$ or $\mathbf{Z}$ as well.

The above theory of missing data has been introduced into recommendation models to address the data biases caused by data missingness (data imbalance). RAPMF [125] first introduces the missing data theory to build the explicit response models and unify the response models with PMF to establish the response aware probabilistic matrix factorization framework. Subsequently, MF-MNAR [77] is proposed based on the missing data theory and probabilistic matrix factorization for MNAR data. Inspired by MF-MNAR, SPMF-MNAR [29] further incorporates social information to model social influence on selection biases. These models achieve desirable performance in debiasing recommendation models and verify the effectiveness of the missing data theory to MNAR modeling.

3.4 Discrete Fourier Transform

Discrete Fourier transform (DFT) is essential in digital signal processing and widely used in signal processing applications [98]. In this paper, we only consider the 1D DFT

that plays a crucial role in our FGCF. Given a sequence of N numbers $\{x_n\}_{n=0}^{N-1}$, the 1D DFT converts the sequence into the frequency domain by:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i(2\pi/N)kn}, \quad k = 0, 1, \cdots, N-1 \tag{3.15}$$

where i is the imaginary unit. DFT generates a new N-length representation each of which X_k denotes the spectrum of the sequence $\{x_n\}_{n=0}^{N-1}$ in the frequency domain. In addition, DFT is an one-to-one transformation. Accordingly, we can recover the original sequence $\{x_n\}_{n=0}^{N-1}$ from its spectrum $\{X_k\}_{k=0}^{N-1}$ by the reverse DFT (IDFT):

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k e^{i(2\pi/N)kn}. \tag{3.16}$$

Considering the symmetry and periodicity properties of $e^{-i(2\pi/N)}$, the *fast Fourier transform* (FFT) [56] is developed to compute the DFT and reduces the complexity of DFT from $O(N2)$ to $O(NlogN)$. Similarly, IDFT can also be computed efficiently via the inverse fast Fourier transform (IFFT). Using FFT and IFFT, one can efficiently perform convolutions over sequences in the time space according to the convolution theory for discrete sequences: considering two sequences $\{g[n]\}$ and $\{h[n]\}$:

$$\mathscr{F}\{g * h\}(f) = (\mathscr{F}\{g\} \circ \mathscr{F}\{h\})(f), f \in \mathbb{R} \tag{3.17}$$

where $(g*h)[n] = \sum_{m=-\infty}^{+\infty} g[m]h[n-m]$ denotes the convolution of g and h, $\mathscr{F}$ denotes the FFT, $\circ$ is the pointwise product and f is the frequency. The convolution theorem states that the Fourier transform of a convolution of two sequences equals to the pointwise product of their Fourier transforms.

3.5 Mutual Information

The mutual information is used to describe the general correlation between variables [166], and the definition is as follows:

$$I(X;Y) = \int_X \int_Y P(X,Y) log \frac{P(X,Y)}{P(X)P(Y)} \tag{3.18}$$

where X, Y are two variables and $P(X), P(Y), P(X,Y)$ are probability distribution and joint probability distribution respectively. According to the chain rule of information, the multivariate mutual information can be defined as follows:

$$I(X_{1:N}; Z) = \sum_{s \subseteq S} I(\{s \cup Z\}), |s| \geq 1 \tag{3.19}$$

where $S = \{X_1, X_2, ..., X_N\}$, s is the subset of S and Z is the target variable.

3.6 Evaluation Metrics

In this section, we introduce all metrics adopted in the experimental evaluation of this thesis, including the metrics for rating estimate, ranking, classification, and informativeness.

3.6.1 Rating Estimate Metrics

To measure the accuracy of rating prediction and time series forecasting, we utilized the most widely used evaluation metrics, namely Mean Absolute Error (MAE) and Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.20}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.21}$$

where n denotes the number of samples in the testing set, $\hat{y}_i$ and y_i denote the prediction and ground truth respectively.

3.6.2 Ranking Metrics

The most common way to assess the recommendation/retrieval performance is to measure whether relevant items are placed in the top positions of a recommendation or retrieval list. Intuitively, ranking metrics are suitable for such application scenarios and have been widely employed to evaluate the recommendation/retrieval performance.

Specifically, given a top-N ranked item set $\hat{R}_N$ and the target ground-truth item set R, Prec@N is calculated as follows:

$$Prec@N = \frac{|R \cap \hat{R}_N|}{N}. \tag{3.22}$$

HR@K is calculated as:

$$HR@K = \frac{|R \cap \hat{R}_N|}{|R|}. \tag{3.23}$$

MAP@N is calculated via the mean of the average precision (AP@N) on all cases, and AP@N is defined by:

$$AP@N = \frac{\sum_{i=1}^{N} Prec@i \times rel(i)}{\min(|R|, N)}, \tag{3.24}$$

where $rel(i)$ equals 1 if $i \in R$, otherwise 0.

3.6.3 Classification Metrics

In classification tasks, we not only evaluate the accuracy of a classifier also investigate its robustness to unbalanced label data. Therefore, the metrics, i.e., accuracy and AUC (area under the ROC curve), are employed to evaluate classification performance. Specifically, the accuracy is calculated as follows:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.25}$$

where TP, FP, TN, and FN denote the number of true positive, false positive, true negative, and false negative respective.

AUC measures the probability that the rank of relevant movies $\mathbf{M}^+$ is higher than irrelevant movies $\mathbf{M}^-$ w.r.t. a group, and it is estimated as follows:

$$AUC = \frac{\sum_{i \in \mathbf{M}^+} \sum_{k \in \mathbf{M}^-} \delta\left[rank(i) < rank(k)\right]}{|\mathbf{M}^+| \bullet |\mathbf{M}^-|} \tag{3.26}$$

where $\delta(.)$ returns 1 if $rank(i) < rank(k)$ and 0 otherwise.

For multi-class problems, AUC is calculated as below:

$$AUC = \frac{2}{|L| \times (|L| - 1)} \sum_{i<j} \frac{A_{ij} + A_{ji}}{2} \tag{3.27}$$

where $|L|$ denotes the number of class labels, and A_{ij} and A_{ji} are the AUC values calculated by considering only cases from classes i and j.

3.6.4 Informativeness Metrics

Mutual Information Neural Estimation (MINE) [10] is proposed to evaluate the informativeness of the learned representation. MINE achieves the estimation of mutual information between high dimensional continuous random variables by gradient descent over neural networks. Specifically, we can design a Mutual Information Neural Estimator (MINE) that is linearly scalable in dimensionality as well as in sample size, trainable through back-prop, and strongly consistent, and then adopt the estimator to calculate the mutual information between learned representation and original data, i.e., evaluating the the informativeness of the learned representation.

Part II

Non-IID Recommender Systems

CHAPTER 4

SEQUENTIAL RECOMMENDATION BY MODELING PREFERENCE DYNAMICS AND FEATURE COUPLINGS

4.1 Introduction

Sequential recommender systems (SRSs) should cater for both stationary and dynamic user preferences. Figure 4.1 illustrates the sequential recommendation of movies by considering the couplings among the user's age, movie genres and current date and the relations between movie series. In Figure 4.1, a user under 18 years old watched a series of movies (labeled by movie genres). Lower green arrows denote the time points when the movies were watched, and the upper lines indicate the couplings between user feature 'Under 18 years old' and movie genres where the thickness of lines refers to the strength of user/item feature couplings. At the current time (the last one in the figure), the film *Home Alone 3* may be highly recommended to the user after its release since (1) the user likely prefers children's and comedy movies, and (2) there is a sequential evolution from *Home Alone 1* to *Home Alone 3*.

When explicit user/item features are available, modeling their feature couplings [139] may disclose their driving roles on user preferences on items and improve recommendation effectiveness and comprehensibility [276]. Existing approaches such as factorization machines [215], statistical recommendation learning [51] and DeepFM [67] embed explicit features into a low-rank latent space and factorize user-item interactions (e.g., ratings and clicks) and feature couplings in terms of the pairwise inner product of

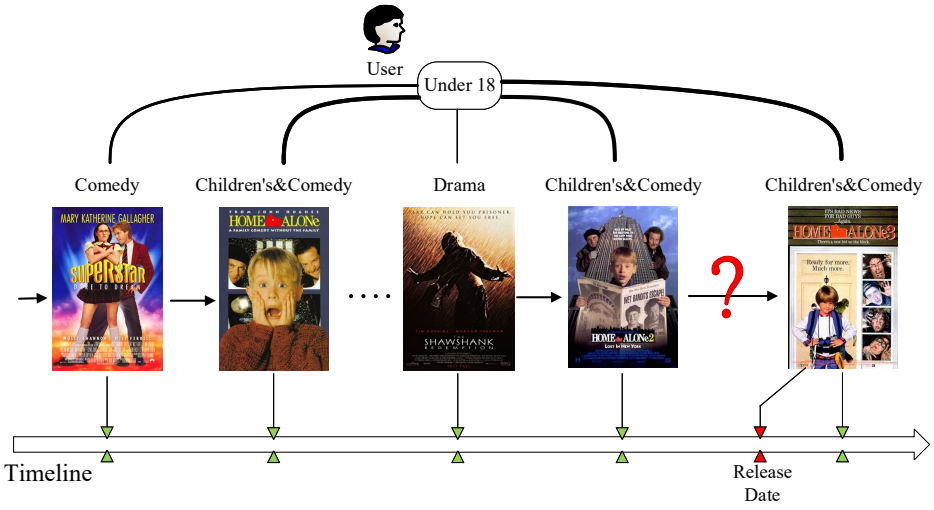


Figure 4.1: An example to illustrate the motivation of user's action on movies.

the embedded latent vectors or latent variable relation modeling. Involving more features (especially categorical features which are converted to one-hot representations) often makes input feature vectors highly sparse [75] and the computation costly especially on recommendation data that is usually sparse and 'long-tailed' as new items or users are continually introduced [51]. It is extremely challenging to model user-item interactions and feature couplings on large, sparse and evolving data [51, 203].

While modeling explicit feature couplings is helpful for disclosing some factors driving user-item interactions and user stationary preferences on items, SRSs have to also consider the time influence [40, 240] on user preferences and contextual change. In reality, recommendation data such as user's clicks, check-ins or e-commerce transactions are stamped with their occurrence timeframes. Such time information reflects user action sequencing and preference dynamics [153, 203, 229, 236], which should be combined with the above user stationary preferences modeling for a more comprehensive understanding and interpretability of user preference formation and evolution [57] and informing recommendation transition.

Incorporating time dynamics into typical temporal and sequential models is reflected in recent work such as Markov chain-based models [74, 172], recurrent neural networks (RNNs) [219, 240], convolutional neural networks (CNNs) [205, 251], and attention networks [96, 269]. While these models capture transitional patterns of user action sequences and achieve state-of-the-art performance in user preference modeling, there are still several issues in user action sequences worthy of further exploration. First,

time-specific information may not be involved in the modeling, instead they rely on temporal order. Second, multi-fold sequential patterns such as union-level and skip-behavior patterns [205] may exist in sequential user actions, where Markov chain-based and RNN-based models are not suitable. Further, user sequential actions take place in their corresponding temporal context which influence future actions [262, 282]. Lastly, explicit user/item feature couplings not only disclose driving factors of user stationary preferences [1, 276] they also supplement sequential modeling e.g. in alleviating the cold-start issues.

Accordingly, neural Time-aware Recommendation Networks (TARN) are introduced to jointly consider user stationary preferences and preference dynamics for sequential recommendation by involving time-specific information, explicit feature couplings, and user action sequences. Specifically, we utilize a feature interaction network to represent user stationary preferences by factorizing the pairwise couplings between the non-zero features of users, items and context through the inner product of their feature embeddings, which can alleviate the data sparsity issues. By assuming user preference dynamics is attributed to the temporal dynamics within a user's sequential actions and is strongly related to the user's recent action sequence, we introduce a convolutional layer with multiple filter widths to capture multi-fold sequential patterns by taking the advantage of CNNs in extracting multi-granular local features [59, 102]. We further propose a temporal action embedding to represent user actions, which combines the embeddings of items and temporal context as the input of the convolutional network to make the network aware of the temporal context.

4.2 Problem Formulation

Here, we introduce the TARN design which incorporates a convolutional network to learn the temporal dynamics of user preferences and a feature factor model to capture user stationary preferences. As shown in Figure 4.2, TARN consists of four components: embedding layers, convolutional network, feature interaction layers, and output layers. The N-size action sequence $\mathscr{S}^t_{u,1:N}$ of user u before time t is fed into the embedding layers and then the convolutional layers to generate the representation of temporal dynamics. The feature vector $\mathbf{X}^t_{u,i}$ containing user features $\mathbf{X}_u$, item features $\mathbf{X}_i$ and temporal context $\mathbf{C}_t$ is fed to the feature interaction network to represent the interactions of user/item/temporal features. These two aspects of representations are combined in the output layers to predict the action at time t. The right two sub components show the

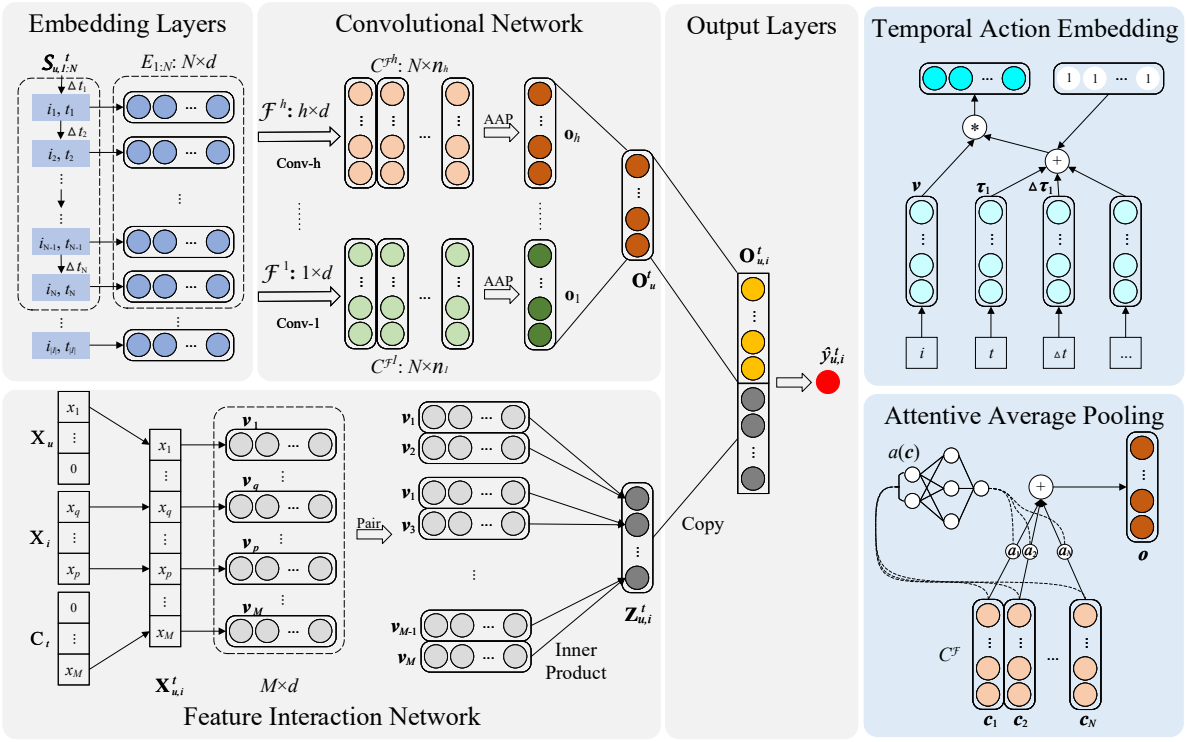


Figure 4.2: The TARN architecture for the time-aware modeling of user-item interactions by involving user action sequences and user/item/temporal feature couplings.

action embedding method and the pooling strategy used in the embedding layers and convolutional network respectively. TARN models the collaborative effect of user/item features, sequential actions, and the temporal context on user actions on items by fusing the three aspects of information to generate sequential recommendations.

Formally, let $\mathscr{I}$ be the set of items, $\mathscr{U}$ be the set of users, and $\mathscr{X}_I$ and $\mathscr{X}_U$ are the corresponding explicit item feature matrix and user feature matrix, respectively. Each user is associated with a sequence of historical actions:

$$\mathscr{S}_u = \{(i_1, t_1), (i_2, t_2), \cdots, (i_{|\mathscr{S}_u|}, t_{|\mathscr{S}_u|})\}, \tag{4.1}$$

where each action (i,t) indicates that item $i \in \mathscr{I}$ is addressed at timestamp t. Assuming that a user's action at a time is influenced by his/her prior actions, we apply a N-sized window sliding over the user's action sequence to construct a clipped action sequence set. We generate the clipped sequence set for a user u as follows:

$$\{\mathscr{S}^{N+1}_{u,1:N}, \mathscr{S}^{N+2}_{u,1:N}, \cdots, \mathscr{S}^{|\mathscr{S}_u|}_{u,1:N}\}. \tag{4.2}$$

$\mathscr{S}^{p}_{u,1:N} = \{(i_{p-N}, t_{p-N}), (i_{p-N+1}, t_{p-N+1}), \cdots, (i_{p-1}, t_{p-1})\}$ is the clipped sequence denoting the N successive actions of user u before time step p. Given the item feature matrix $\mathscr{X}_I$,

the feature vector $X_u \in \mathscr{X}_U$, and the clipped sequence $\mathscr{S}^t_{u,1:N}$ of user u, the recommendation problem is to suggest new items to user $u \in \mathscr{U}$ at time step t with those items that are not in $\mathscr{S}_u$ but are likely to be favored by the user.

4.3 Time-aware Recommendation Networks

4.3.1 Modeling Temporal Dynamics

Leveraging the advantages of CNNs in capturing local features and relations [59, 97], our method applies a convolutional layer with multi-width filters to discover multi-fold sequential patterns of user actions on items. First, we introduce the proposed temporal action embedding.

4.3.1.1 Temporal Action Embedding

To make the convolutional network sensitive to a temporal context, we introduce a temporal action embedding method to represent user actions. It embeds item variables, the time of the current action, and the time interval between the current action and the previous action in the same low-dimensional space and combines these embeddings as the representation of user sequential actions, i.e., the input of the convolutional layer in TARN.

First, we utilize time encoding schemes to transform a standard timestamp to a unique time id for embedding. Specifically, the schemes are customized per application scenarios to extract time factors, e.g., hour, day and weekday types, from a timestamp. To choose a proper time encoding scheme, we analyze the relationships between user actions with different time factors and try different schemes to test their performance[1]. Since the schemes cannot extract all the time factors and do not completely depict the temporal shift between successive actions, we further consider the time interval between successive actions which can be interpreted as the period of user absence from the recommender system. The time interval is calculated as follows, where t_j is the timestamp at j-th action:

$$\Delta t_j = \lfloor log(t_j - t_{j-1} + 1) \rfloor. \tag{4.3}$$

Here, Δt_j reflects how long a user has been absent from the recommender system at the j-th action ($\Delta t_1 = \max_j \Delta t_j$). Assuming that the influence of the time interval Δt_j

[1]We extract hour for MovieLens and Last.fm and day types (weekend and weekday) for Tafeng in the experiments after trying different schemes.

becomes small with the time interval being larger, we use the logarithmic function to rescale the time interval and adopt floor function ($\lfloor\cdot\rfloor$) to convert a scaled time interval to a positive integer for embedding. Note that the converted integers have an upper bound and are enumerable since timestamp t_j is bounded by the max timestamp in each application.

After encoding the timestamps and transferring time intervals into integers, we embed these two aspects of time information and combine their resultant embeddings with item embeddings to obtain temporal action representation as shown in the Temporal Action Embedding in Figure 4.2. Specifically, given any N-action sequence clipped at time step t of user u, i.e., $\mathscr{S}^t_{u,1:N} = \{(i_1,t_1),(i_2,t_2),\cdots,(i_N,t_N)\}$, the sequence is represented by the following embedding matrix (for concision, we omit subscript u and time step t in the following):

$$\begin{aligned} E_{1:N} &= \{\mathbf{e}_1,\mathbf{e}_2,\cdots,\mathbf{e}_N\}^T \in \mathbb{R}^{N\times d}, \\ s.t.\, \mathbf{e}_j &= \nu_j * (\tau_j + \Delta\tau_j + \mathbf{1}),\ j \in \{1,2,\cdots,N\}, \end{aligned} \tag{4.4}$$

where $\mathbf{e}_j$ denotes the embedding vector for j-th action. $\nu_j,\tau_j,\Delta\tau_j \in \mathbb{R}^{d\times 1}$ are the embedding vectors for item i_j, timestamp t_j and time interval Δt_j respectively, and $*$ denotes the element-wise product. Inspired by the idea of initializing context embeddings in [12], we initialize τ and $\Delta\tau$ by 0-mean Gaussian and thus treat the context term $(\tau_j + \Delta\tau_j + \mathbf{1})$ as a mask over item embedding. Note that the proposed temporal action embedding can easily be extended to integrate other contextual information such as location by summing all context embeddings in the multiplicative item in Equation (4.4).

4.3.1.2 Multi-width Convolutional Network

By assuming that a user's action at time t is influenced by his/her recent actions, we represent the previous action sequence to model the temporal dynamics of user preferences. Specifically, we feed $N \times d$ embedding matrix $E_{1:N}$ of the N-action sequence into the convolutional layers. Herein, we utilize d-height convolutional filters to slide over the embedding matrix to capture the action-level local feature combinations. Considering the existence of multi-fold sequential patterns, we leverage a convolutional layer with multiple filter widths which is formulated as follows (shown in the convolutional network in Figure 4.2):

$$\begin{aligned} \mathscr{F} &= \mathscr{F}^1 \cup \mathscr{F}^2 \cup \cdots \cup \mathscr{F}^H, \\ \mathscr{F}^h &= \{f^l \in \mathbb{R}^{h\times d} | l \in \{1,\cdots,n_h\}\}, h \in \{1,\cdots,H\} \end{aligned} \tag{4.5}$$

where H is the maximum filter width, h is the width of a filter and n_h denotes the number of filters with a width h. Intuitively, $\mathscr{F}^h$ denotes the set of h width filters, and its l-th filter f^l treats the embedding vector of each action as a whole and covers h actions each step during convolutional calculation. Thus, given the embedding $E_{1:N}$, the j-th convolution feature by the wide convolution[2] of filter $f^l \in \mathscr{F}^h$ is given by:

$$c_j^{h,l} = \phi_a(f^l \cdot E_{j:j+h-1} + b_c),\ s.t.\ j \in \{1,2\cdots,n_h\}. \tag{4.6}$$

Here, operator $\cdot$ computes the inner-product, b_c is a bias term, and ϕ_a is a non-linear activation function ($ReLU$ in TARN). $E_{1:N}$ is extended to $E_{1:N+h-1}$ with 0 padding for wide convolution when calculating $c_j^{h,l}$. The filter f^l slides along the embedding matrix $E_{1:N}$ to produce a feature map: $C^{h,l} = [c_1^{h,l} c_2^{h,l} \cdots c_N^{h,l}]$ for the l-th filter in $\mathscr{F}^h$. To differentiate different granular features, we stack the convolutional outputs extracted by filters with the same width h, i.e., $\mathscr{F}^h$:

$$C^{\mathscr{F}^h} = \begin{bmatrix} C^{h,1} \\ C^{h,2} \\ \vdots \\ C^{h,n_h} \end{bmatrix}^T = \begin{bmatrix} c_1^{h,1} c_1^{h,2} \cdots c_1^{h,n_h} \\ c_2^{h,1} c_2^{h,2} \cdots c_2^{h,n_h} \\ \vdots \\ c_N^{h,1} c_N^{h,2} \cdots c_N^{h,n_h} \end{bmatrix}. \tag{4.7}$$

Consequently, for all the filters in $\mathscr{F}$, the convolutional output over the embedding matrix $E_{1:N}$ is,

$$C^{\mathscr{F}} = \left[C^{\mathscr{F}^1} C^{\mathscr{F}^2} \cdots C^{\mathscr{F}^H} \right]. \tag{4.8}$$

4.3.1.3 Attentive Average Pooling

We introduce an **A**ttentive **A**verage **P**ooling (AAP) over each stacked convolutional output, i.e., $C^{\mathscr{F}^h}$, to capture the large-span feature combinations shown in Figure 4.2. Additionally, the attentive pooling learns to distribute high weights to useful features, facilitating effective information passing to high-level networks.

Specifically, we denote $C^{\mathscr{F}^h} = [\mathbf{c}_1 \mathbf{c}_2 \cdots \mathbf{c}_N]^T$ for concision where $\mathbf{c}_j \in \mathbb{R}^{n_h \times 1}$ represents the j-th row in matrix $C^{\mathscr{F}^h}$. Obviously, $\mathbf{c}_j$ consists of the convolutional outputs of all filters $f \in \mathscr{F}^h$ at j-th position (action). The weight of each position $j \in \{1,2\cdots,N\}$ is formulated as follows:

$$a_{hj} = \frac{exp(\mathbf{L}^T ReLU(\mathbf{W}_a \mathbf{c}_j + \mathbf{b}_a)}{\sum_{\mathbf{c}_j \in C^h} exp(\mathbf{L}^T ReLU(\mathbf{W}_a \mathbf{c}_j + \mathbf{b}_a))}, \tag{4.9}$$

[2]Wide convolution guarantees the same heights of convolutional input and output and retains more feature combinations near the end of the sequence.

where $\mathbf{W}_a \in \mathbb{R}^{d_a \times n_h}$, $\mathbf{L} \in \mathbb{R}^{d_a \times 1}$ and $\mathbf{b}_a \in \mathbb{R}^{d_a \times 1}$ are identical for each $C^{\mathscr{F}^h}$ for simplicity and convenience for training, meaning that all convolutional outputs share attention parameters. The attentive average pooling over C_h is built upon the weighted sum of $\mathbf{c}_j$:

$$\mathbf{o}_h = \sum_{j=1}^{N} a_{hj} \mathbf{c}_j^T, s.t. \sum_{j=1}^{N} a_{hj} = 1, o_h \in \mathbb{R}^{n_h \times 1}. \tag{4.10}$$

For any $C^{\mathscr{F}^h} \subset C^{\mathscr{F}}$, we obtain a corresponding new representation $\mathbf{o}_h$ by the attentive average pooling. We then concatenate the series of $\mathbf{o}_h$ and feed them into a fully-connected neural network to further extract high-level features:

$$\mathbf{O} = \sigma_o(\mathbf{W}_o \begin{bmatrix} \mathbf{o}_1 \\ \mathbf{o}_2 \\ \vdots \\ \mathbf{o}_H \end{bmatrix} + \mathbf{b}_o), \tag{4.11}$$

where $W_o \in \mathbb{R}^{d \times (H * n_h)}$ is a weight matrix, $\mathbf{b}_o \in \mathbb{R}^{d \times 1}$ is the corresponding bias term and σ_o is the activation function (specifically, $ReLU$ is used in experiments). $\mathbf{O}$ as the final output of the convolutional layers stands for the sequence representation reflecting user preference dynamics. Correspondingly, we denote the sequence representations of $\mathscr{S}^t_{u,1:N}$ with $\mathbf{O}^t_u$ which reflects the preference dynamics of user u at time step t. The vector representing the temporal dynamics of user preferences is fed into a concatenation layer along with the feature coupling vector, the details of which are discussed in Section 4.3.3.

4.3.2 Modeling Feature Couplings

Recent studies demonstrate that feed-forward neural networks are inefficient in capturing feature couplings [12]. Inspired by obtaining the second-order couplings in FM, we calculate the feature couplings between explicit features by the inner-product of the corresponding feature embedding vectors as shown in the feature interaction network in Figure 4.2. Since the feature vector is highly sparse, we only consider the pairwise couplings between non-zero feature vectors. In addition, contextual information also interacts with user/item features, which finally influences user actions. For this reason, we combine the embedded user, item and context features into a vector and model their coupling relationships.

Given a user u and an item i at time step t, $\mathbf{X}^t_{u,i} = [x_1, \cdots, 0, x_q, \cdots, x_p, 0, \cdots, x_M]$ denotes the concatenated feature vectors of user u ($\mathbf{X}_u \in \mathscr{X}_U$), item i ($\mathbf{X}_i \in \mathscr{X}_I$), and temporal context at timestamp t ($t, \Delta t \in \mathbf{C}_t$), shown in Figure 4.2. Note that the feature

interaction network is flexible so other context features such as location are easily incorporated in the network by concatenating the feature vectors of other contexts with $\mathbf{X}^t_{u,i}$. In $\mathbf{X}^t_{u,i}$, categorical features are converted to binary features by one-hot encoding while numerical features remain unchanged. Furthermore, M denotes the number of original categorical and numerical features. We then calculate the pairwise non-zero feature couplings as follows:

$$\begin{aligned} \mathbf{Z}^t_{u,i} &= [z_{1,2}, z_{1,3}, \cdots, z_{M-1,M}]^T, \\ s.t. \quad z_{p,q} &= x_p x_q \mathbf{v}_p \cdot \mathbf{v}_q,\ p < q,\ x_p, x_q \in \mathbf{X}^t_{u,i}, \end{aligned} \tag{4.12}$$

where $\mathbf{v}_p, \mathbf{v}_q \in \mathbf{V}$ denotes the d-dimension position embedding vectors of the p-th and q-th feature within $\mathbf{X}^t_{u,i}$, and $\mathbf{V} \in \mathbb{R}^{|\mathbf{X}^t_{u,i}| \times d}$ denotes the position embedding matrix. Intuitively, value $z_{p,q}$ measures the correlations between the p-th feature and the q-th feature, and $\mathbf{Z}^t_{u,i}$ captures the correlations between the user, item and context features in the granularity of their pairwise (second-order) couplings. Both the inputs of the convolutional network and feature interaction network (i.e., $\mathcal{S}^t_{u,1:N}$ and $\mathbf{X}^t_{u,i}$) involve the same temporal information. We thus use shared embeddings for the temporal information.

Different from common FM-based models such as DeepFM, NFM and Deep&Wide [36] which involve both first-order and second-order (pairwise) feature couplings, we are only interested in pairwise feature relations, which is demonstrated to be adequate and effective as shown in Section 4.4.6.1. Additionally, the high-order feature relations adopted in recent models like NFM and DCN [227] can model more complex feature relations to improve model performance to some extent. However, introducing high-order feature couplings incurs a high computational burden or makes it difficult to interpret the couplings between features. To this end, we simplify the whole model and only adopt the second-order feature couplings.

In addition, not all feature couplings are equally constructive, and useless couplings may introduce noise and degrade the performance. It is crucial to flexibly distribute weights for different feature couplings. Instead of using the attention mechanism as in AFM [250], we copy the feature coupling vector $\mathbf{Z}^t_{u,i}$ to the output layer in the integration with the convolutional output in Equation (4.13) and learn the weights directly, which makes our model more flexible than the aforementioned FM-based methods.

4.3.3 Prediction and Inference

We combine the sequence representation and feature couplings for prediction. Before feeding the combined vector into the output layers as shown in Figure 4.2, we multiply

$\mathbf{O}_u^t$ with an item-related vector to specify the sequence representation for each particular item and then stack the resultant vector with the feature coupling vector, which is formulated as:

$$\hat{y}_{u,i}^t = \sigma(\mathbf{w}^T \begin{bmatrix} \mathbf{O}_{ui}^t \\ \mathbf{Z}_{ui}^t \end{bmatrix} + b) = \sigma(\mathbf{w}^T \begin{bmatrix} \mathbf{O}_u^t \circ \widetilde{\mathbf{v}}_i \\ \mathbf{Z}_{ui}^t \end{bmatrix} + b), \tag{4.13}$$

where $\widetilde{\mathbf{v}}_i \in \widetilde{\mathbf{V}} \in \mathbb{R}^{|\mathscr{I}| \times d}$ is a transformation vector specified for item i, $\mathbf{w} \in \mathbb{R}^{(d+|\mathbf{Z}_{ui}|)\times 1}$ and $b \in \mathbb{R}$ are the output weight vector and bias respectively. Operator $\circ$ denotes the element-wise product and $\sigma(\cdot)$ is the Sigmoid function. $\hat{y}_{u,i}^t$ predicts the probability of a target item i for user u at time step t based on the sequence representation $\mathbf{O}_u^t$ and feature coupling vector $\mathbf{Z}_{ui}^t$. Thus, we denote:

$$p(i|\mathscr{S}_{u,1:N}^t, \mathscr{X}_U, \mathscr{X}_I) = \hat{y}_{u,i}^t \tag{4.14}$$

To train the model, an N-width window slides over the action sequence of each user u (padding 0 is performed if the length is smaller than N) to generate a training sequence set, i.e., the clipped sequence set $\{\mathscr{S}_{u,1:N}^1, \mathscr{S}_{u,1:N}^2, \cdots, \mathscr{S}_{u,1:N}^T\}$, and the corresponding next predictive action (precisely item) set of each clipped sequence $\mathscr{S}_{u,1:N}^t$ is treated as $\mathscr{I}_t^u$. We denote the collection of time steps to predict $\mathscr{T}^u = \{1, 2, \cdots, T\}$ for each user u. The likelihood of all the sequences in the training set is given by:

$$p(\mathscr{S}|\Theta, \Omega) = \prod_{u \in U} \prod_{t \in \mathscr{T}_u} \prod_{i \in \mathscr{I}_t^u} \hat{y}_{u,i}^t \prod_{\bar{i} \notin \mathscr{I}_t^u} (1 - \hat{y}_{u,\bar{i}}^t). \tag{4.15}$$

Taking the negative logarithm of the likelihood, we have the objective function w.r.t. the cross entropy loss:

$$\mathscr{L}_{\Theta,\Omega} = \sum_{u \in \mathscr{U}} \sum_{t \in \mathscr{T}^u} \sum_{i \in \mathscr{I}_t^u} -log(\hat{y}_{u,i}^t) + \sum_{\bar{i} \notin \mathscr{I}_t^u} -log(1 - \hat{y}_{u,\bar{i}}^t), \tag{4.16}$$

where $\bar{i}$ is a sampled negative item (in the experiment we adopt the commonly-used random sampling for efficient training) and $\Theta = \{\mathbf{V}, \nu, \tau, \Delta\tau, \mathbf{W}_a, \mathbf{W}_o, \mathbf{w}, \mathbf{L}, \widetilde{\mathbf{V}}, \mathbf{b_a}, \mathbf{b_o}, b\}$ are model parameters learned by minimizing the objective function. Furthermore, the hyper-parameters $\Omega = \{d, N, H, n_h, d_a\}$ are selected via empirical experiments and grid search. After all the parameters are learned, Equation (4.13) is used to calculate the probability for all items, and the items with the top-K highest probability are recommended to users.

Equation (4.13) shows that the prediction is made on top of two aspects of information: the previous action sequence and the user, item and context-combined features. Actually,

by treating the convolutional output as a 'temporal bias' (preference dynamics), user actions are reflected in the stationary feature couplings. Let us rewrite Equation (4.13):

$$y_i^{u,t} = \sigma(\begin{bmatrix} \mathbf{w}_o^T & \mathbf{w}_Z^T \end{bmatrix} \begin{bmatrix} \mathbf{O} \circ \widetilde{\mathbf{v}}_i \\ \mathbf{Z}_{ui} \end{bmatrix} + b) = \sigma(\begin{bmatrix} \mathbf{e}^T & \mathbf{w}_Z^T \end{bmatrix} \begin{bmatrix} \mathbf{O} \circ \widetilde{\mathbf{v}}_i \\ \mathbf{Z}_{ui} \end{bmatrix} + b), \tag{4.17}$$

where $\mathbf{W}_o$ and $\mathbf{W}_Z$ reflect the significance of the two aspects in prediction and $\mathbf{e}$ denotes a unit vector. Equation (4.17) is obtained since we have $\mathbf{w}_o^T(\mathbf{O} \circ \widetilde{\mathbf{v}}_i) = \mathbf{e}^T(\mathbf{O} \circ \widetilde{\mathbf{v}}_i \circ \mathbf{w}_o) = \mathbf{e}^T(\mathbf{O} \circ \widetilde{\mathbf{v}}_i)$ and parameter $\mathbf{w}_o$ can be absorbed in $\widetilde{\mathbf{v}}_i$. By leaving the 'temporal bias' apart, $\mathbf{w}_Z$ is helpful to investigate and interpret the feature-based couplings between users, items and context.

Complexity analysis. To make a recommendation for each user u at time step t, the proposed model calculates $\hat{y}_{u,i}^t$ for all items $i \notin \mathcal{S}_u$ and recommends the items with the top-k highest predictive probabilities. Hence, the complexity for making recommendation for all users is $O(|\mathcal{U}||\mathcal{I}|P + k|\mathcal{U}||\mathcal{I}|)$, where U is the user set, $\mathcal{I}$ is the item set, k is the number of recommended items, and P denotes the time complexity of calculating $\hat{y}_{u,i}^t$. Since a user's previous action sequence for a given time step is fixed, the output of the convolutional network can be calculated once and used to calculate $\hat{y}_{u,i}^t$ on all items, which means that the complexity of convolutional operations can be ignored relative to the calculation of feature couplings. The complexity of calculating pairwise feature couplings is $O(Md)$ for each item [215] where M denotes the number of features and d is the dimension of latent factors. Accordingly, for all users, the total complexity of recommending top-k items is:

$$O(|\mathcal{U}||\mathcal{I}|P + k|\mathcal{U}||\mathcal{I}|) = O(|U||\mathcal{I}|Md + k|\mathcal{U}||\mathcal{I}|).$$

Practically, recommendation is made based on a candidate item set (denoted as $\mathcal{C}$) rather than all items $\mathcal{U}$ to save time while the corresponding time complexity is $O(|U||\mathcal{C}|Md + k|U||\mathcal{C}|)$ and $|\mathcal{C}| \ll |\mathcal{I}|$. Similar to other neural recommenders, TARN can be trained offline, thus we do not investigate its training efficiency in these experiments.

4.4 Experiments and Evaluation

We perform extensive experiments on three public datasets to investigate the following research problems:

Q1 What is the effect of integrating feature couplings and preference dynamics on TARN-enabled recommendation?

Table 4.1: Statistics of three recommendation data.

Dataset	**Item#**	**User#**	**Interaction#**	**Feature#**
MovieLens	3,706	6,040	1,000,209	10,089
Tafeng	23,071	12,889	657,211	38,004
Last.fm	174,903	983	14,248,823	217,321

Q2 How does TARN perform in comparison with the state-of-the-art recommendation methods?

Q3 How effective is TARN in capturing user preference dynamics compared with the state-of-the-art methods?

Q4 Does modeling explicit feature couplings enable TARN to handle the cold-start issue?

4.4.1 Experimental Settings

4.4.1.1 Datasets

The effect of our method in capturing user preference dynamics and feature couplings is tested on three publicly available datasets. Table 4.1 summarizes the statistics of the datasets after pre-processing them.

MovieLens[3]. This dataset collects the ratings from users who joined MovieLens in 2000. As this work concerns feature couplings, user demographics (i.e., gender, age and occupation) and item information (i.e., genre) are selected to train the models. We then convert each rating application (i.e., a pair of user ID and movie ID) along with user/item side information and temporal context into multi-hot representations, resulting in $10,089$ features in total. All the ratings in MovieLens are scaled from $\{1,2,3,4,5\}$ to a target value 1 to indicate a user has rated a movie.

Tafeng[4]. This dataset contains Chinese grocery store transactional data from November 2000 to February 2001. Customer demographics (i.e., customer ID, age and pin code) and item attributes (i.e., original ID, sub class, amount, asset and price) are used in the experiments. Numeric attributes (e.g., amount, asset and price) are discretized, and each transaction (i.e., a pair of customer ID and original ID) along with customer and item

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

[4]https://www.kaggle.com/chiranjivdas09/ta-feng-grocery-dataset

information and temporal context is converted to a feature vector, resulting in 38,004 features in total.

Last.fm[5]. This dataset records the music listening habits of nearly 1,000 users till May, 5th 2009. User information (i.e., user ID, gender, age and country) and music information (i.e., track ID and artist ID) are used as features. Regarding all the above categorical features, each record is converted to a 217,321-sized feature vector for model training.

Since all the datasets contain only positive instances, to ensure model generalization in such one-class settings, for each positive instance of a user, we randomly sample two negative instances that have no interactions (i.e. no ratings or no purchase) with the user. All the negative samples are assigned with a target value of 0.

4.4.1.2 Evaluation Metrics

To evaluate the performance in the sequence settings, we employ the widely used leave-one out protocol. Specifically, we hold the latest item (action) in each user's item sequence in the test set and the remaining items in the training set. After a model is trained, we generate, for each user, a personalized ranking of items that have no interaction with the user in the training set, and we evaluate the performance on the recommended ranking in terms of two ranking-based metrics [205, 283]: HR@K and mean average precision (MAP).

4.4.1.3 Baselines

To investigate the design effectiveness of TARN, our model is customized into the following versions:

- **T-MP**: This replaces the attentive average pooling in TARN with **M**ax **P**ooling in order to investigate the effectiveness of the attentive average pooling. Max pooling is performed over the convolutional output of each filter, i.e., each column of $C^{\mathscr{F}^h}$.

- **T-IF**: This models user preference dynamics over user action sequences by a convolutional network with an **I**dentical-width **F**ilter in contrast to the multi-filter design in TARN. In the experiments, we fix the width $h = 4$.

[5]https://www.dtic.upf.edu/ ocelma/MusicRecommendationDataset/

- **T-NT**: This does **N**ot involve any **T**emporal context information in modeling user preference dynamics, which verifies the contribution made by the temporal action embedding.

To make a fair comparison, our experiments use the same settings on other parameters of the variants of TARN. Furthermore, we compare TARN with the following baselines:

- **POP**: This ranks items based on their popularity determined by the number of user interactions on them in the training set. It is a non-personalized baseline.
- **FM** [171]: This learns both first- and second-order feature relations in the latent feature space.
- **NFM** [75]: This combines the linearity of FM and the non-linearity of neural networks to jointly model low-order and high-order feature relations and achieves state-of-the-art performance in sparse prediction.
- **RRN** [240]: This is the state-of-the-art rating prediction method with sequential settings by introducing LSTM and a traditional low-rank factorization to capture user preference dynamics and user-item interactions.
- **Caser** [205]: This is a CNN-based model for sequential recommendation by leveraging a CNN with multiple horizontal convolutional filters and one vertical convolutional filter to capture sequential patterns and combining user embeddings to capture user global preferences.
- **SASRec** [96]: A state-of-the-art self-attention-based recommendation model uses an attention mechanism to capture the long-term semantics of action prediction.
- **SITAR** [169]: This is the latest context-aware sequential recommendation model by involving the temporal context into stacked RNNs to capture user preference dynamics that vary with contextual dynamics and temporal gaps.
- **RCNN** [251]: This is the latest RNN- and CNN-based model with RNN to capture complex long-term dependencies and CNN to extract short-term sequential patterns on the recurrent hidden states.

These state-of-the-art methods are deliberately chosen for the following considerations: FM and NFM verify the contributions of capturing the dynamics of user preferences; RRN and SITAR are RNN-based methods; and Caser and RCNN are CNN-based

methods. These methods are selected to justify the effectiveness of incorporating feature couplings and to compare RNNs and CNNs in capturing temporal dynamics. Furthermore, SASRec is selected to compare with the typical attention-based method. Markov chain-based methods such as in [74, 172] are not compared since the latest deep neural network-based methods outperform these methods [205].

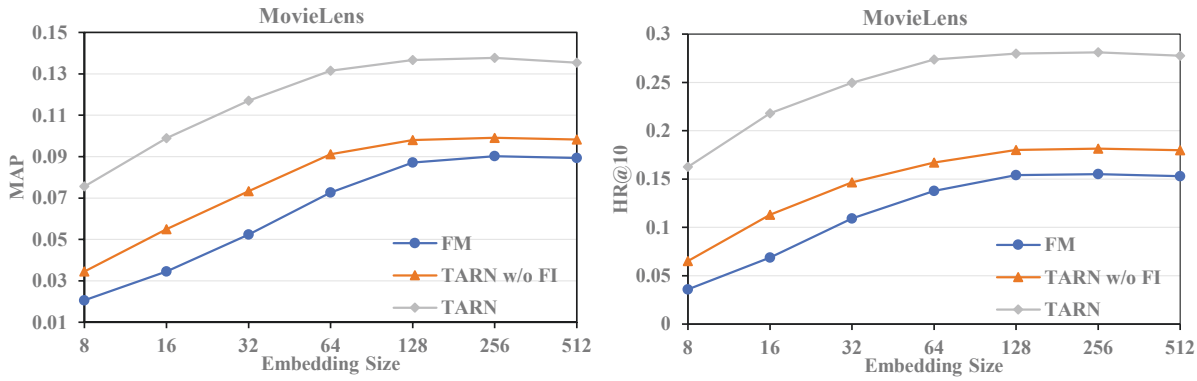
4.4.1.4 Implementation and Parameter Settings

We implement TARN, its variants, RRN and CRNN in TensorFlow. FM is based on LibFM [215]. NFM, Caser, SASRec and SITAR are derived from their GitHub versions released by their authors. Furthermore, we perform a grid search of the learning rate over $\{1e^{-4}, 1e^{-3}, \ldots, 1e^{-1}\}$ and L_2 regularization rate over $\{1e^{-6}, 1e^{-5}, \ldots, 1e^{-1}\}$ to select the best performance for FM and NFM and obtain the parameter settings for RRN, Caser, SASRec, SITAR and RCNN according to their source codes and recommended settings. To tune the hyperparameters in TARN, we leave out the penultimate actions in each user's sequence for validation. Finally, we set 1) the embedding size: 64; 2) convolutional layer: $n_h = 16$ and $H = 4$, meaning for each $h \in [1,4]$, we have 16 filters; 3) attentive average pooling: $d_a = 32$; 4) optimizer: Adam [103]; (5) batch size: 256; and (6) learning rate: $1e^{-3}$. For a fair comparison and efficiency, all methods are set with an embedding size of 64 if not specified, although a larger embedding size may perform better.

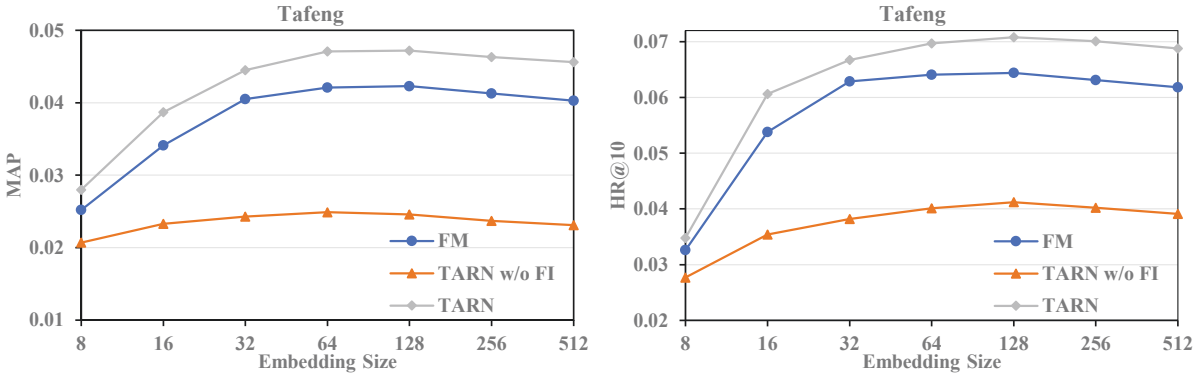
4.4.2 Ablation Study

To analyze the effectiveness of integrating feature couplings with preference dynamics, we report the ablation test of TARN under embedding sizes of $\{8, 16, 32, 64, 128, 256, 512\}$. Specifically, we discard the feature interaction network in Figure 4.2 to test the contribution of modeling feature couplings and compare this with FM to test the performance gain derived by capturing user preference dynamics.

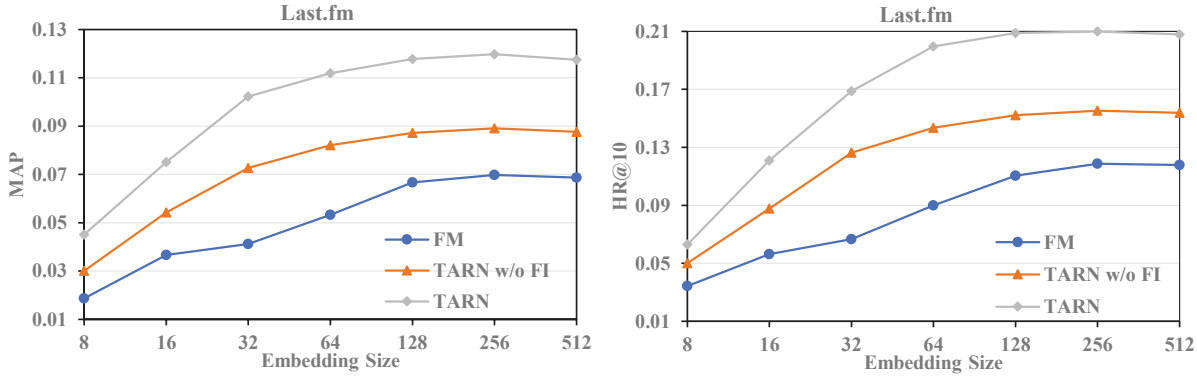
We test HR@10 and MAP of these methods w.r.t. different embedding sizes. The results are shown in Figure 4.3 where "TARN w/o FI" denotes TARN without the feature interaction network. Obviously, TARN outperforms the other two methods, indicating that both feature couplings and user preference dynamics contribute to the prediction. The two comparative methods have advantages in different datasets. Specifically, TARN w/o FI achieves significant performance improvement with an increase of embedding size and outperforms FM on MovieLens and Last.fm, but it achieves a small improvement on Tafeng. In contrast, FM performs better when embedding size increases and outperforms



(a) MovieLens



(b) Tafeng



(c) Last.fm

Figure 4.3: Ablation test performance: HR@10 and MAP of FM, TARN w/o feature couplings and TARN w.r.t. embedding sizes.

TARN w/o FI on Tafeng. The results confirm the fact that user preferences for media (e.g., movies and music) are dynamic and subject to the current context, while users in Tafeng have stable actions and preferences for grocery. This shows that simultaneously modeling user preference dynamics and feature couplings is necessary to deeply understand the

Table 4.2: TARN recommendation performance comparison with baselines on MovieLens, Tafeng and Last.fm.

Dataset	Metric	POP	FM	NFM	RRN	Caser	SASRec	SITAR	CRNN
MovieLens	MAP	0.0201	0.0727	0.1028	0.1138	0.1231	<u>0.1283</u>	0.1198	0.1275
	HR@10	0.0363	0.1379	0.1889	0.2402	0.2627	<u>0.2692</u>	0.2531	0.2689
Tafeng	MAP	0.0284	0.0421	<u>0.0435</u>	0.0334	0.0326	0.0391	0.0407	0.0389
	HR@10	0.0427	0.0629	<u>0.0647</u>	0.0461	0.0472	0.0566	0.0582	0.0554
Last.fm	MAP	0.0044	0.0533	0.0672	0.0952	0.0989	0.1029	0.0973	<u>0.1037</u>
	HR@10	0.0051	0.0899	0.1139	0.1782	0.1818	0.1907	0.1791	<u>0.1911</u>

Dataset	Metric	T-MP	T-IF	T-NT	TARN	Imp.
MovieLens	MAP	0.1269	0.1119	0.126	**0.1315**	2.5%
	HR@10	0.2695	0.2397	0.2664	**0.2738**	1.7%
Tafeng	MAP	0.0484	0.0437	0.0463	**0.0471**	8.3%
	HR@10	0.0694	0.0651	0.0687	**0.0697**	7.7%
Last.fm	MAP	0.1052	0.0991	0.1058	**0.1109**	6.9%
	HR@10	0.1948	0.185	0.1951	**0.1997**	4.5%

intrinsic relations between the user, item and context features and the user action sequential patterns to enable powerful recommendations.

With an increasing embedding size, TARN achieves a more stable performance than the other two methods, and it outperforms the other two methods consistently. The results indicate that 1) a larger embedding size is beneficial to improve performance due to the increased modeling capability; 2) intrinsically, feature couplings and preference dynamics can work collaboratively and complement each other; 3) the design of TARN effectively captures and integrates the two aspects for the prediction. Furthermore, the performance worsens when the embedding size becomes large, which is attributed to the fact that complex models easily overfit training data.

4.4.3 Performance Comparison

We investigate the recommendation performance of TARN against the state-of-the-art baselines w.r.t. HR@10 and MAP. The results are summarized in Table 4.2, where the best result in each row is highlighted in bold and the best baseline results are underlined for each dataset. Table 4.2 enables the following key observations.

First, Table 4.2 shows that TARN achieves the best results. Specifically, TARN

achieves an improvement of 1.7% and 2.5% in terms of HR@10 and MAP over SASRec on MovieLens where SASRec has the best performance among the state-of-the-art methods, and it also performs better than the baselines on Tafeng; it especially outperforms NFM, the best baseline on Tafeng, by up to 7.7% and 8.3% in terms of HR@10 and MAP. Furthermore, on Last.fm, TARN respectively exhibits an improvement of 4.5% and 6.9% in terms of HR@10 and MAP over the state-of-the-art CRNN. The results indicate that TARN achieves state-of-the-art performance.

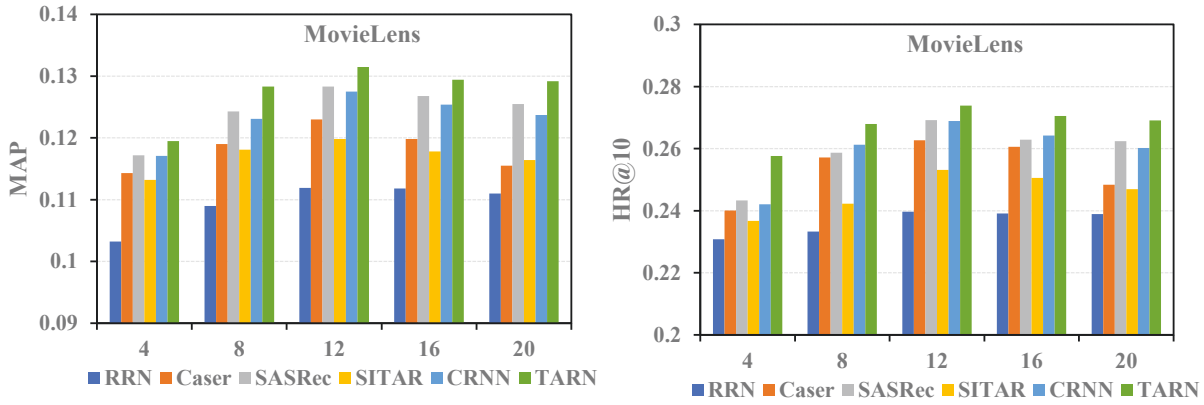
Second, the aforementioned result verifies the effectiveness of integrating feature couplings and user preference dynamics. Methods like FM, NFM, SASRec and CRNN which only involve either of the two aspects cannot fully capture user preferences and degrade the recommendation performance. Similarly, RRN and Caser fail to exploit the available explicit features and do not work well on a dataset with strong feature couplings, e.g., Tafeng. SITAR performs well on Tafeng but obtains bad results on the other two, indicating our design of interaction layers and convolutional layers is more effective to capture context influence and sequential patterns.

Further, compared to RNN- and CNN-based methods, the variants of TARN, i.e., T-IF, T-MP and T-NT, also achieve desirable and even better performance. This improvement is attributed to modeling the coupling relationships between explicit features. The results confirm that explicit features may be coupled to drive user preferences and action on an item[1].

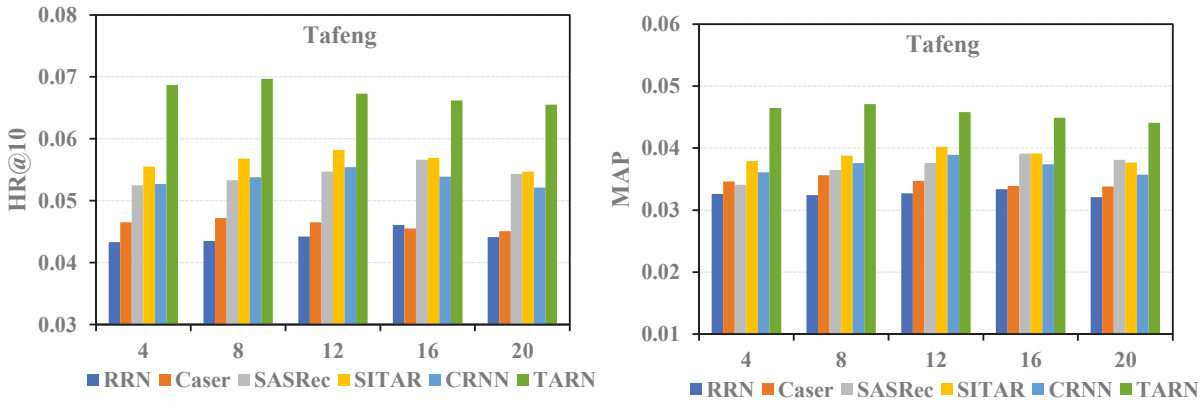
Lastly, Table 4.2 shows that TARN works better than T-NT, T-IF and T-MP as expected, indicating that the proposed temporal action embedding, the setting of multi-width filters, and the attentive average pooling contribute to improving recommendation performance. Specifically, the multi-width filter design facilitates the extraction of different granularities of features and is suitable for capturing multi-fold sequential patterns. The attentive average pooling is beneficial to extract more information and learn a more effective aggregation (i.e., weighted average) in the setting of pooling over the sequence. T-NT achieves the best performance among the three variants, indicating temporal context information contributes to model user-item interactions but it is not the most important fact compared with user/item features and sequential patterns.

4.4.4 Influence of Sequence Length

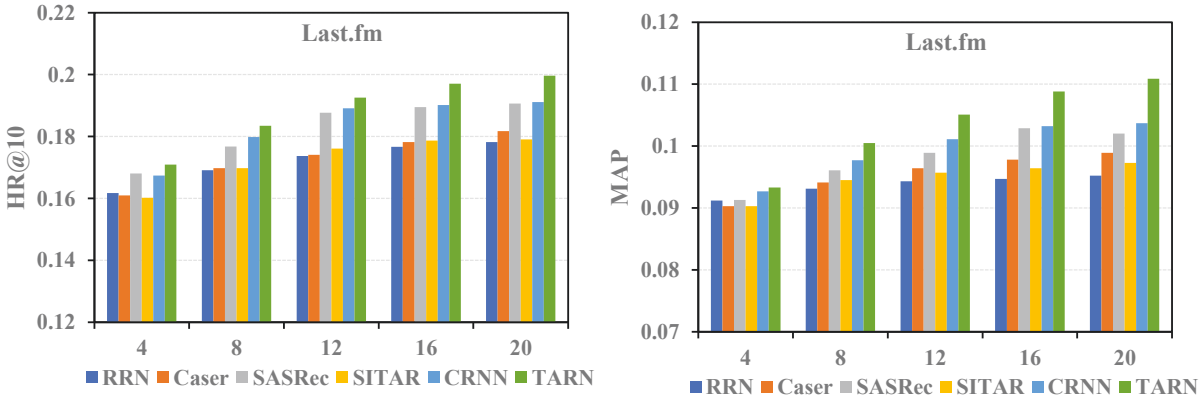
To further verify TARN's capability in modeling user preference dynamics, we compare RRN, Caser, SASRec, SITAR, CRNN and TARN under different clipped sequence lengths



(a) MovieLens



(b) Tafeng



(c) Last.fm

Figure 4.4: HR@10 and MAP comparison over different sequence lengths between TARN and the sequential baselines.

N. The comparison provides a comprehensive observation of sequential recommendation using these methods. The results in terms of MAP and HR@10 are shown in Figure 4.4.

Figure 4.4 reports that TARN achieves higher MAP than the baselines over different sequence lengths. This result is partially attributable to the contribution of capturing feature couplings and demonstrates the positive effect of integrating feature couplings and preference dynamics. Except on Tafeng where user preference dynamics are not obvious, the performance of the other methods is strongly influenced by sequence length. Specifically, the best sequence length for the methods are 12 and 20 on MovieLens and Last.fm respectively (note that RRN is less influenced by sequence length since RRN embeds all previous actions at each step for input to capture global information and is more stable). The results show that the comparative methods achieve their best performance under different sequence lengths on two datasets, which is intuitively reasonable listening to music may occur more frequently than watching movies and there exist stronger long-term dependencies within music sequence data. It is worth noting that TARN works much better and more stably than Caser, which demonstrates again that attentive average pooling is effective and more useful to 'summarize' the convolutional output. Furthermore, we observe that all methods perform worse on MovieLens when the sequence length becomes larger than 12. This may be because noise (irrelevant actions) is introduced with an increase in sequence length.

4.4.5 Cold-start Test

To investigate the way TARN handles cold-start users, we compare TARN to the sequential baselines RRN, Caser, SASRec, SITAR and CRNN on users with few training samples. The results are shown in Table 4.3 which reports the average HR@1 over ten experiments and the intervals ($[a, b]$) in the table header denote those users with the number of training samples in $[a, b]$. Tafeng is not included since RRN and Caser perform badly on Tafeng. From Table 4.3, we can make the following observations: 1) TARN achieves obviously higher hit rates (highlighted in bold) over the baselines even for users who have few training samples. 2) SITAR performs slightly better than the other baselines. 3) Caser and RRN are comparable with SASRec and CRNN on users with quite a few samples, and SASRec and CRNN perform worst. This is because both TARN and SITAR involve context features into the preference modeling, but TARN considers the explicit features such as user profile and item features and captures the couplings between user, item and context features. However, Caser and RRN model user stationary preferences by factorizing only user-item interactions in a latent space as matrix factorization does but do not consider explicit features, and SASRec and CRNN do not even consider user stationary preferences but strongly rely on user action sequences.

Table 4.3: Cold-start test of TARN over the sequential baselines in terms of HR@1.

Dataset	Method	[1,5]	[6,10]	[11,20]
MovieLens	RRN	0.0313 ±0.0039	0.0464 ±0.003	0.0481 ±0.0032
	Caser	0.0342 ±0.0032	0.0497 ±0.0028	0.0536 ±0.0029
	SASRec	0.0288 ±0.0021	0.0414 ±0.0029	0.0485 ±0.0035
	SITAR	0.0352 ±0.003	0.0514 ±0.0033	0.0561 ±0.0035
	CRNN	0.0282 ±0.0031	0.0397 ±0.0023	0.0476 ±0.0025
	TARN	**0.0408** ±0.0038	**0.0574** ±0.0039	**0.0615** ±0.0045
Last.fm	RRN	0.0287 ±0.0024	0.0453 ±0.0032	0.0454 ±0.0027
	Caser	0.0302 ±0.0026	0.0449 ±0.0029	0.0461 ±0.0033
	SASRec	0.0264 ±0.0026	0.0418 ±0.0031	0.0443 ±0.0032
	SITAR	0.0311 ±0.0032	0.0464 ±0.0029	0.0491 ±0.0034
	CRNN	0.0252 ±0.0031	0.0437 ±0.0024	0.0456 ±0.0029
	TARN	**0.0408** ±0.0038	**0.0574** ±0.0039	**0.0615** ±0.0045

The results indicate that TARN achieves better performance than the state-of-the-art methods in the cold-start settings and demonstrates that modeling explicit feature couplings is beneficial to capture user stationary preferences in predicting user actions.

4.4.6 Visualization and Interpretability

In this section, we explore the rationale and interpretability of our model and discuss interesting findings from the model. Due to space limitations, we only analyse the visualization on MovieLens as an example and the results on Tafeng and Last.fm show the same findings.

4.4.6.1 Explicit Feature Couplings

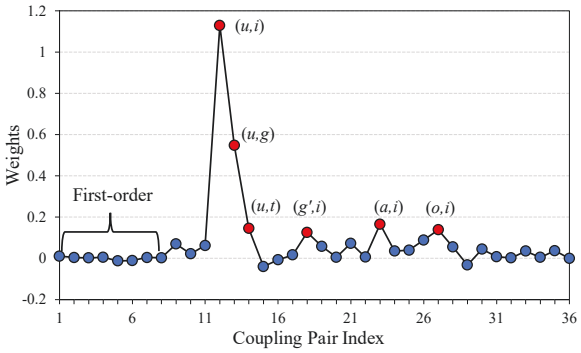
To investigate the predictive effects of couplings between user, item and context features, we design the following experiment: following FM in capturing both first-order and second-order feature couplings as DeepFM [67], NFM [75] and AFM [250], we involve the first-order information into the feature interaction network by modifying Equation (4.12) as follows:

$$\begin{aligned} \mathbf{Z}_{u,i}^{t} &= [z_{0,1}, z_{0,2}, \cdots, z_{M-1,M}]^{T}, \\ s.t. \quad z_{p,q} &= x_p x_q \mathbf{v}_p \cdot \mathbf{v}_q,\ p < q,\ x_p, x_q \in \widetilde{\mathbf{X}}_{u,i}^{t}. \end{aligned} \tag{4.18}$$

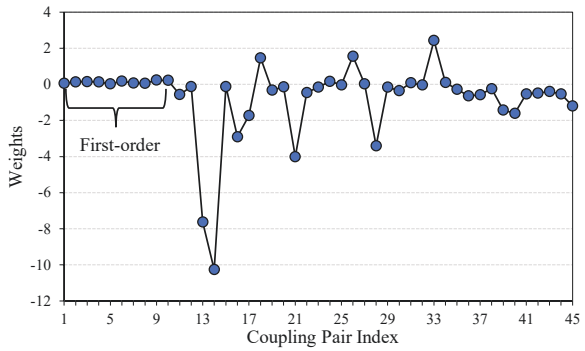
Here, $\widetilde{\mathbf{X}}_{u,i}^{t} = [x_0, x_1, \cdots, 0, x_q, \cdots, x_p, 0, \cdots, x_M]$ denotes the extended feature vector from $\mathbf{X}_{u,i}^{t}$ where the additional feature $x_0 = 1$ is a constant and the corresponding embedding vector $\mathbf{v}_0 = \mathbf{e}$ is a unit vector. Such a model is trained till convergence on the same parameter settings as TARN, and we visualize the well-trained output weights on the feature coupling vector, i.e., W^{Z} in Equation (4.17), to reveal the contribution of first-order and second-order feature couplings in the setting of our model. The absolute values of the resultant weights are high, meaning the corresponding feature couplings have an effect on the prediction.

The weight visualization on MovieLens is shown in Figure 4.5 where the y-axis denotes the weight values and the x-axis denotes the coupling pair indices calculated by a mapping function $\Phi : z_{p,q} \mapsto \mathbb{R}$ and $\Phi(z_{p,q}) = p + q$ where $z_{p,q} \in \mathbf{Z}_{u,i}^{t}$. In Figure 4.5, we use MovieLens as an example and number the first-order feature couplings from 1 to 8 and the second-order feature couplings larger than 8. There are a total of 36 couplings resulting from the 8 features in MovieLens. Figure 4.5 shows that the learned weights for all the first-order couplings are around 0 and most of the weights for the second-order couplings are positive. This observation indicates that the first-order couplings adopted in methods like NFM and DeepFM do not contribute to the prediction of user actions in our case. In other words, there is no need to consider the first-order features, and the pairwise feature couplings are suitable and sufficient to capture the coupling relations between user, item and context features and model user stationary preferences.

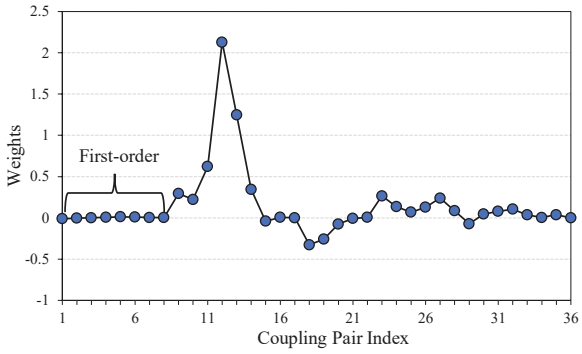
In addition, we select six feature pairs (highlighted by the red nodes) whose weights are higher than those of the other pairs. Here, we denote u for user ID, i for item ID, g for genre of items, t for timestamp, g' for gender, a for age and o for occupation. The six feature pairs are : (user ID, item ID), (user ID, genre), (user ID, time), (gender, item ID), (age, item ID), and (occupation, item ID), which represent inter-feature couplings [293] between users, items and time context features. In contrast, the intra-feature couplings



(a) MovieLens



(b) Tafeng



(c) Last.fm

Figure 4.5: Visualization of weight for MovieLens, Tafeng and last.fm.

[293] within user features, item features and context features have relatively small weights. These findings indicate that inter-feature couplings between users, items and context are more productive to the prediction.

In the observation, the coupling pair (user ID,item ID) has the largest weight, which is equivalent to factorizing user-item interactions as MF does. This is reasonable because user ID and item ID are representative and informative. Furthermore, user/item features

are often partially available and cannot reflect the comprehensive relations between users and items. In addition, the pairs (user ID, genre) and (user ID and time) have higher weights, indicating that the genre of a movie and the temporal context are influential to user actions as expected. Similarly, the other red nodes reflect that the corresponding feature pairs have an impact on the prediction.

4.4.6.2 Preference Dynamics

To further investigate the intuition of modeling user preference dynamics, Figure 4.6 illustrates the influence of previous items on prediction for four users. In the figure, the histograms show the statistics of attention weights on the latest 20 movies of each selected user. The three highest weighted movies (left of the red arrow) and the target movie (right of the red arrow) are listed below which are labelled with their genres. The figure shows the collected attention weights (y-axis) on each position (corresponding to the x-axis, denoting the position of items in a user's action sequence) collected from TARN with $H = 1$ and $n_h = 32$ on the MovieLens dataset. Note that a higher attention weight on an item reflects the item is more influential to the prediction of the target item. We then present the three highest weighted items and the target item and investigate the influence of user preference dynamics in terms of explicit item feature movie genres.

From Figure 4.6, we observe that previous movies are assigned different weights according to the attention mechanism, which is acceptable since user actions are differently influenced by previous actions, e.g., watching series movies. In addition, we observe that the most influential (highest weighted) movies are of similar/same genre as the target movies. The results demonstrate that the learned attention weights are meaningful and explainable and show that users prefer similar movies (e.g., movies of the same genre) in the short term. The 4-th selected case is exceptional which is explainable since the user action on movies 'Drama' and 'Thriller' are determined by user stationary preferences for 'Drama' or 'Thriller' although the previous action sequence presents an obvious pattern of choosing 'Action' and 'Adventure' movies.

4.5 Conclusions

In this chapter, we aim to understand how the explicit features of users, items and context influence user preferences and how user preference dynamics influence user actions over time. Specifically, our main contributions include:

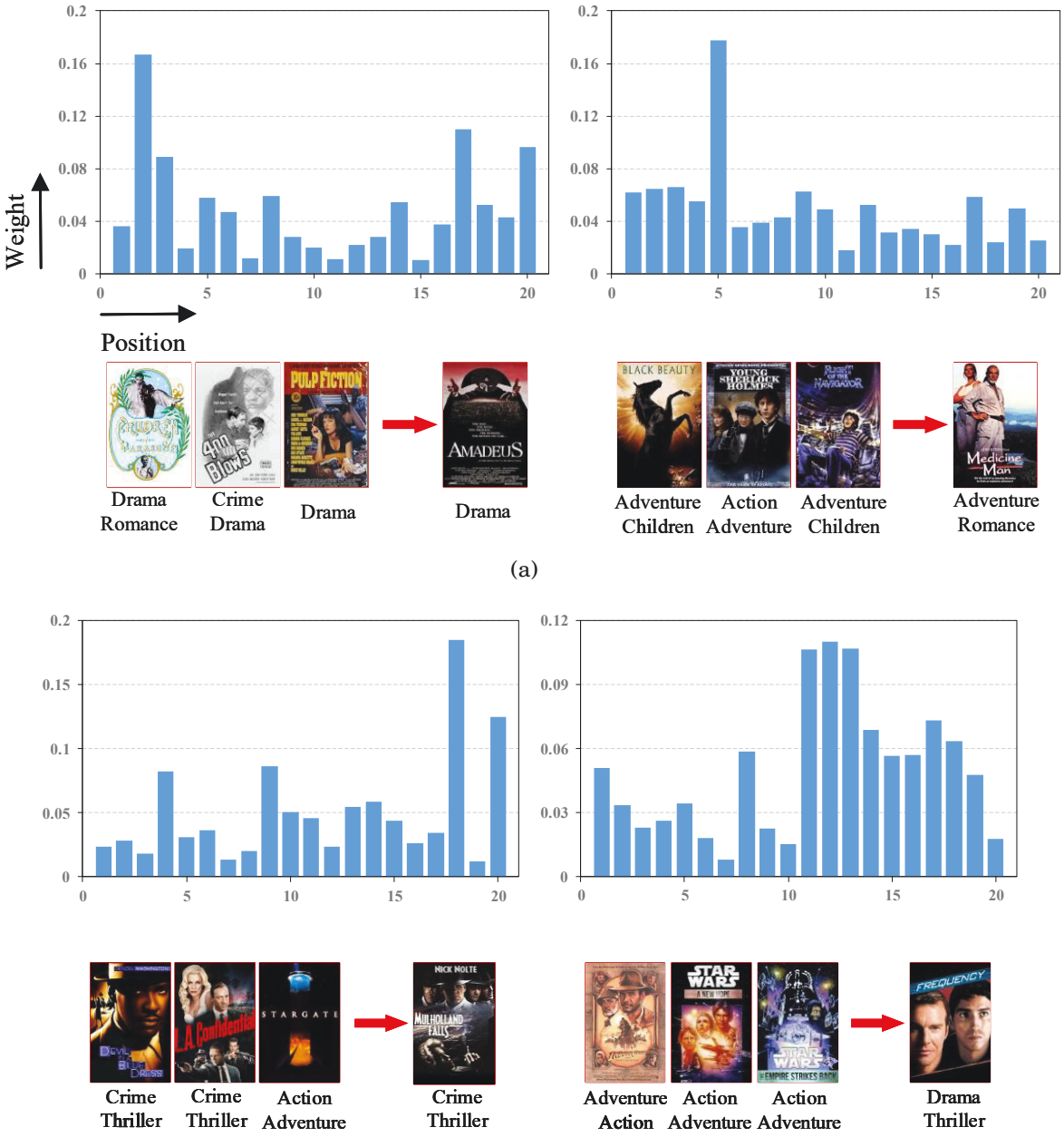


Figure 4.6: Illustration of the influence of preference dynamics on prediction on the MovieLens dataset.

- The proposed TARN jointly models user stationary preferences and preference dynamics for sequential recommendation. Specifically, TARN has a feature factor network to factorize user/item/context feature couplings and a convolutional network fed by user action sequences to model the temporal dynamics of user

preferences.

- To make the convolutional network sensitive to a temporal context, a novel temporal action embedding is proposed to embed items and the temporal context into the same space and represent user actions by combining their embeddings as the inputs of the convolutional network.
- An attentive average pooling is introduced upon the convolutional outputs to obtain significant features and large-span feature combinations, facilitating effective information passing to higher layers.
- We provide visualization to interpret the rationale and the interesting findings of the model.

The empirical results on three real-world datasets show that (1) the proposed mechanisms in TARN for modeling feature couplings and user preference dynamics are effective; (2) TARN outperforms the state-of-the-art methods in terms of various verification aspects; and (3) both the temporal action representation and attentive average pooling have a positive impact on improving recommendation performance.

CHAPTER 5

TRIPARTITE COLLABORATIVE FILTERING FOR RATING DEBIASING ON MISSING-NOT-AT-RANDOM DATA

5.1 Introduction

Generally, rating data in recommendations is missing-not-at-random, that is that the distributions of observed ratings and all ratings are not identical (i.e., heterogenity). Some recent studies further explore the MNAR rating issue [185, 233, 258] to debias the rating estimation. For example, a classic debiasing approach [126] to the MNAR data is the probabilistic theory of missing data. Such methods [29, 77, 125, 146] treat the problem as missing data imputation based on the joint likelihood of the missing rating model and the complete rating model, where the missed ratings (i.e., non-selections) are dependent on the rating values. The intuition behind these methods is that all ratings are firstly generated by the complete rating model and the missing rating model then estimates which entries to be selected (or missed) according to their rating values.

Beyond the dependency on rating values, we argue that the generation process of ratings may be actually more complicated with the MNAR ratings. Revisiting movie recommendation, movie recommenders often suggest those movies that they believe interesting to users, e.g., popular movies, but rarely suggest movies potentially less interesting. Meanwhile, users cannot select and rate those movies unobservable to them. The observability of movies to users influences the user selection of movies. The MNAR perspective indicates the *item observability* to users and the *user selection* of items may

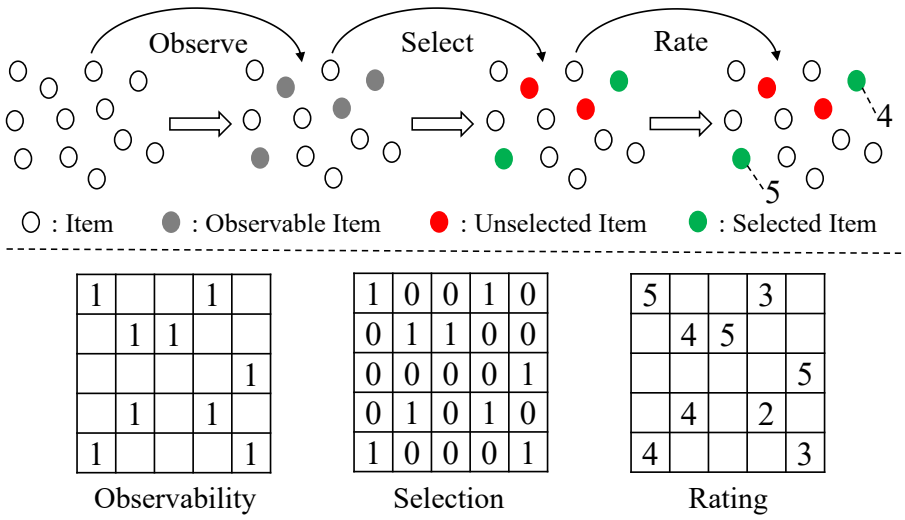


Figure 5.1: The influence of item observability and user selection on the rating generation.

jointly influence the rating generation [8, 149, 261], as shown in Figure 5.1. The figure reflects that the missing ratings contain both preferred yet unobserved entries caused by poor item observability and non-preferred (also called negative) entries. The above debiasing methods neglect the impact of item observability but unreasonably treat all missing entries as non-preferred, which may not conform to the generation process of ratings and lead to a biased modeling of actual user selection and missing ratings.

We argue to simultaneously model item observability and user selection ti debiase rating estimation, which however is challenging especially when only on the rating data as these aspects are often coupled and co-influence each other [1, 157]. Such modeling needs to properly infer the relationships between the triple aspects while excessively complex modeling may render overfitting. To tackle these challenges, we take two new perspectives: (1) ratings are influenced by factors describing user selection; and (2) item observability depicts the scope of user selection and could correct the probability of the missing entries being negative. New CF models built on the two aspects have potential to address the MNAR nature of rating data and avoid modeling to be biased and skewed to the available ratings.

The above aspects motivate us to develop a tripartite collaborative filtering (TCF) framework by incorporating both item observability and user selection into rating estimation to cater for the MNAR rating data and to tackle the rating estimation bias. We further instantiate the framework by Probabilistic Matrix Factorization (PMF) [179] and propose a Tripartite Probabilistic Matrix Factorization model (TPMF) to infer the three

corresponding variables in three sub-models: (1) *a complete rating model* to factorize the ratings with multifaceted factors and model the dependency of ratings on user selection by factorizing the two aspects into shared subspaces simultaneously; (2) *a complete observability model* to introduce a Bernoulli distribution to model the item observability, which determines whether an item is observable to a user and assigns each missing entry a confidence of being truly negative; and (3) *a user selection model* to treat user selection by following a Gaussian distribution whose mean is a function of the corresponding rating value and determining which observable items will be selected by the user.

5.2 Problem Formulation

We are given a rating dataset $\mathscr{D} = \{r_{ij} | 1 \le i \le n, 1 \le j \le m, r_{i,j} \in \{1,2,\cdots,L\}, (i,j) \in \mathscr{A}\}$ of discrete ratings by n users on m items, where $\mathscr{A}$ denotes the set of user-item pairs on which a rating is available. The goal of recommender systems is to estimate ratings for those missing entries, i.e., user-item pair $(i,j) \notin \mathscr{A}$, denoted $\bar{\mathscr{A}}$.

From $\mathscr{D}$ we can obtain triple aspects of rating data: rating $\mathbf{R} \in \mathbb{R}^{n \times m}$, observability $\mathbf{O} \in \{0,1\}^{n \times m}$, and selection $\mathbf{S} \in \{0,1\}^{n \times m}$ as shown in Figure 5.1. Specifically, $\mathbf{R}$ is the rating matrix where $\mathbf{R}_{\mathscr{A}}$ denotes the available ratings. The observability matrix $\mathbf{O}$ is binary and partly available where $o_{ij} = 1$ if the j-th item is observable to the i-th user and $o_{ij} = 0$ otherwise. $\mathbf{S}$ is a binary selection matrix where element $s_{ij} = 1$ denotes that the i-th user has selected the j-th item and $s_{ij} = 0$ denotes the opposite. Naturally, we obtain that: 1) an item is rated (or unrated) by a user if and only if the item is selected (or unselected) by the user, i.e., $s_{ij} \in \mathbf{S}_{\mathscr{A}} \leftrightarrow s_{ij} = 1$ and $s_{ij} \in \mathbf{S}_{\bar{\mathscr{A}}} \leftrightarrow s_{ij} = 0$; 2) an item selected by a user must be observable to the user first, i.e., $p(o_{ij} = 1 | s_{ij} = 1) = 1$ and $o_{ij} \in \mathbf{O}_{\mathscr{A}} \leftrightarrow o_{ij} = 1$; and 3) an item unobservable to a user cannot be selected by the user, i.e., $p(s_{ij} = 0 | o_{ij} = 0) = 1$. Our objective is to build a debiasing model for rating estimation on the MNAR rating data by jointly inferring the above triple aspects.

5.3 Methodology

In this section, we first introduce the problem definition and our proposed tripartite collaborative filtering (TCF) framework for the MNAR data. We then instantiate the TCF into a tripartite probabilistic matrix factorization model (TPMF).

5.3.1 Tripartite Collaborative Filtering Framework

Inspired by the work in [126], we propose a novel tripartite collaborative filtering (TCF) framework for the MNAR rating data by inferring the triple aspects of rating data (e.g., $\mathbf{R}, \mathbf{O}, \mathbf{S}$). In the TCF framework, we propose three sub-models for the triple aspects: 1) a complete rating model (CRM) to predict $\mathbf{R}$ with parameters Ω_r; 2) a complete observability model (COM) to generate $\mathbf{O}$ with parameters Ω_o, and 3) a user selection model (USM) to infer $\mathbf{S}$ with parameters Ω_s. The joint distribution for $\mathbf{R}$, $\mathbf{O}$ and $\mathbf{S}$, given Ω_r, Ω_s and Ω_o, is below:

$$p(\mathbf{R}, \mathbf{O}, \mathbf{S}|\Omega) = p(\mathbf{R}|\Omega_r)p(\mathbf{O}|\Omega_o)p(\mathbf{S}|\mathbf{R}, \mathbf{O}, \Omega_s), \tag{5.1}$$

where $\Omega = \{\Omega_r, \Omega_o, \Omega_s\}$, and Ω_r and Ω_s share a part of parameters. The intuition behind the joint distribution shows: CRM (i.e., $p(\mathbf{R}|\Omega_r)$) first generates ratings for all user-item pairs, and unobservable user-item pairs are then filtered by COM (i.e., $p(\mathbf{O}|\Omega_o)$), finally USM (i.e., $p(\mathbf{S}|\mathbf{R}, \mathbf{O}, \Omega_s)$) determines which observable pairs will be available (i.e., which item is selected by the user). The generation process assumes that all ratings and item observability are foreknown and user's subsequent selection of an item relates to his/her rating value on the item and the item observability to the user. In addition, CRM and USM share a set of parameters to model the multifaceted correlation between user selection and ratings.

This tripartite framework explores the complex dependencies between item observability, user selection, and ratings. The framework is flexible in that we can specify different distributions for each sub-model to satisfy the needs of various real cases, and it is easy to incorporate with metadata via modeling the correlation between the triple aspects of rating data with specific metadata. The constraint is that the dependencies among three sub-models are fixed to guarantee the TCF effectiveness, and CRM and USM should share some parameters to learn the influence of user selection on ratings.

5.3.2 The TPMF Model

Next, we instantiate the TCF framework in terms of probabilistic matrix factorization and propose a Tripartite Probabilistic Matrix Factorization model (TPMF) to infer the triple aspects of rating data by the three sub-models: Complete Observability Model (COM), User Selection Model (USM), and Complete Rating Model (CRM), as shown in Figure 5.2. It contains three sub-models USM, CRM and COM. s_{ij} in dark shade is fully available, r_{ij} and o_{ij} are in light shade where part of the r_{ij} and o_{ij} values are available.

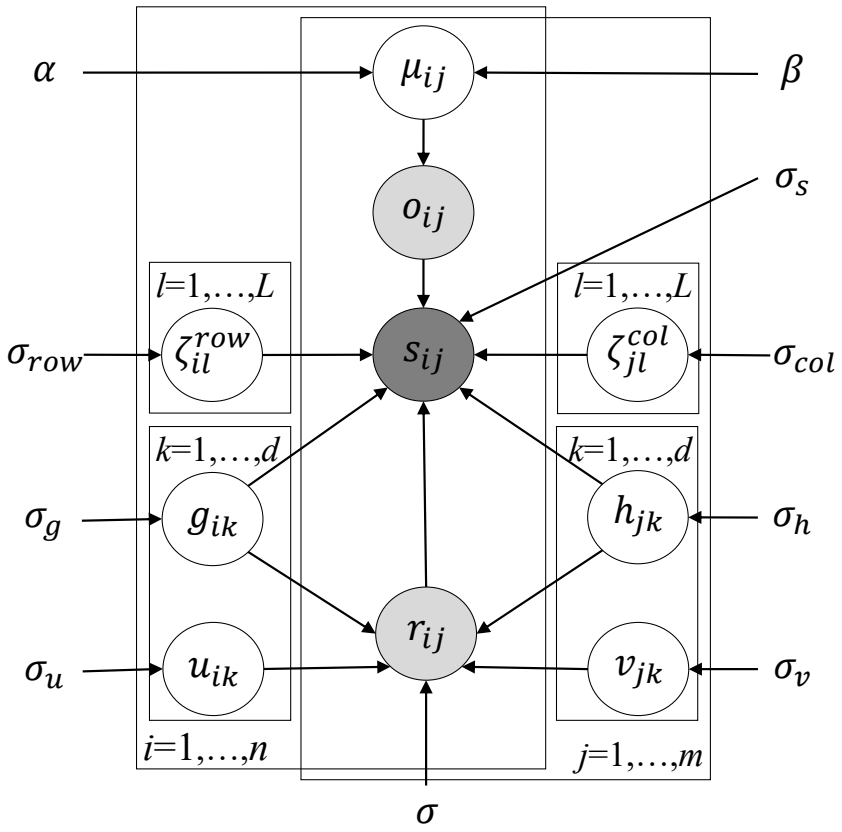


Figure 5.2: Graphical representation of TPMF.

Complete Observability Model (COM). Similar to [122], we assume that the binary **O** follows a Bernoulli distribution whose mean is drawn from a Beta distribution. Specifically, we have:

$$p(\mathbf{O}|\boldsymbol{\mu}) = \prod_{i=1}^{n}\prod_{j=1}^{m} \mathscr{B}(o_{ij}|\mu_{ij}), \tag{5.2}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{n\times m}$ and $\mu_{ij} \in \boldsymbol{\mu}$ denotes the prior probability that item j is observable to user i. For simplicity and avoiding overfitting, we assume the item observability is dependent on item popularity: $\mu_{ij} = \mu_j \sim Beta(\alpha, \beta)$. If extra metadata (e.g., user demographic or item features) is available, it can be used to infer item observability and differentiate item observability for different users.

User Selection Model. We adopt matrix factorization to factorize variable **S** and model the variable as a function of **R** and **O**, see Figure 5.2. Specifically, we treat $s_{ij}|o_{ij} = 0$ following constant distribution (denoted by ρ_0) since we have $p(s_{ij} = 0|o_{ij} = 0) = 1$, and we further model $s_{ij}|o_{ij} = 1$ with a Gaussian distribution (note that the Bernoulli

distribution is also suitable but brings difficulty in inference). Then, we have:

$$p(\mathbf{S}|\mathbf{R},\mathbf{O},\Omega_s) = \prod_{i=1}^{n}\prod_{j=1}^{m} \mathcal{N}(s_{ij}|\hat{s}_{ij},\sigma_s^2)^{o_{ij}} \rho_0^{1-o_{ij}}, \tag{5.3}$$

$$\hat{s}_{ij} = \mathbf{G}_i^T \mathbf{H}_j + \sum_{l=1}^{L} (\boldsymbol{\zeta}_{il}^{row} + \boldsymbol{\zeta}_{jl}^{col}) \mathbb{I}[r_{ij} = l] + b_s, \tag{5.4}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function, b_s is a bias term and Ω_s denotes $\mathbf{G}$, $\mathbf{H}$, $\boldsymbol{\zeta}^{row}$, $\boldsymbol{\zeta}^{col}$ and b_s. Two matrices $\mathbf{G} \in (0,1)^{d\times n}$ and $\mathbf{H} \in (0,1)^{d\times m}$ with $d < \min(n,m)$ are used to factorize $\mathbf{S}$ and follow truncated Gaussian distributions. $\boldsymbol{\zeta}^r \in \mathbb{R}^{n\times L}$ and $\boldsymbol{\zeta}^c \in \mathbb{R}^{m\times L}$ follow zero-mean spherical Gaussian. ζ_{il}^{row} and ζ_{jl}^{col} reflect the influence of rating value r_{ij} on s_{ij}. Intuitively, a larger value of $(\zeta_{il}^{row} + \zeta_{jl}^{col})$ when $r_{ij} = l$ implies a higher probability that $s_{ij} = 1$.

Complete Rating Model (CRM). We further factorize R by the inner product of two low-rank latent matrices $U \in \mathbb{R}^{d\times n}$ and $V \in \mathbb{R}^{d\times m}$, representing latent user preferences and item attraction respectively. Specifically, we assume Gaussian noise on the ratings below:

$$p(\mathbf{R}|\mathbf{U},\mathbf{V},\sigma) = \prod_{i=1}^{n}\prod_{j=1}^{m} \mathcal{N}(r_{ij}|\mathbf{U}_i^T\mathbf{V}_j,\sigma^2), \tag{5.5}$$

where $\mathbf{U}$ and $\mathbf{V}$ follow a zero-mean spherical Gaussian distribution. However, in addition to the influence of rating values on user selection, it is worthy noting how a user selection affects the user rating. We expect that user ratings are also influenced by the factors describing user selection. To model the factor correlation, we regularize the factorization of $\mathbf{R}$:

$$p(\mathbf{R}|\mathbf{U},\mathbf{V},\mathbf{G},\mathbf{H},\sigma) = \prod_{i=1}^{n}\prod_{j=1}^{m} \mathcal{N}(r_{ij}|\hat{r}_{ij},\sigma^2), \tag{5.6}$$

where the estimated rating $\hat{r}_{ij} = \mathbf{U}_i^T \boldsymbol{\Gamma}_{ij} \mathbf{V}_j$ and we have:

$$\boldsymbol{\Gamma}_{ij} = \frac{d\left[diag(\mathbf{G}_i \circ \mathbf{H}_j) + \varepsilon \mathbf{I}\right]}{\mathbf{G}_i^T\mathbf{H}_j + d\varepsilon}, \tag{5.7}$$

where d is the latent dimension, operator $\circ$ calculates the element-wise product and $diag(\cdot)$ denotes a function constructing a diagonal matrix with a vector. $0 \le \varepsilon \le 1$ is an adjustment factor that large ε reduces the influence of $\mathbf{G}_i$ and $\mathbf{H}_j$ on $\hat{r}_{ij}$ and avoids the denominator being zero.

Let $\mathbf{\Gamma}_{ij}^{kk}$ be the k-th ($k \in [1,d]$) diagonal element, we have $\frac{d\varepsilon}{d-1+\varepsilon} < \mathbf{\Gamma}_{ij}^{kk} < \frac{d+d\varepsilon}{1+d\varepsilon}$ and $\mathbb{E}_k(\mathbf{\Gamma}_{ij}^{kk}) = \mathbb{E}_k(\frac{dg_{ik}h_{jk}+d\varepsilon}{\mathbf{G}_i^T\mathbf{H}_j+d\varepsilon}) = 1$. Hence, we can treat $\mathbf{\Gamma}_{ij}$ as a mask over the d multiplicative factors in calculating $\mathbf{U}_i^T\mathbf{V}_j$, and Equation (5.6) is equivalent to PMF when $\mathbf{\Gamma}_{ij}$ equals an identity matrix. A larger value of $g_{ik}h_{jk}$ contributes more to user selection, and it also upweights $u_{ik}v_{jk}$ in the estimation of user ratings. The above settings constrain that user preference and item feature show consistency to some extent on the estimation of user selection and rating.

Joint Model. Based on the three sub-models, we obtain the following log joint probability according to Equation (5.1):

$$\begin{aligned} & log(\mathbf{R},\mathbf{O},\mathbf{S}|\Omega_o,\Omega_s,\Omega_r) \\ =& \sum_{i=1}^{n}\sum_{j=1}^{m} o_{ij} log\mathcal{N}(s_{ij}|\hat{s}_{ij},\sigma_s^2) + log\mathcal{N}(r_{ij}|\hat{r}_{ij},\sigma^2) \\ & + log\mathcal{B}(o_{ij}|\mu_{ij}) + (1-o_{ij})log\mathbb{I}(s_{ij}=0) + \mathscr{C} \end{aligned} \tag{5.8}$$

where $\mathscr{C}$ denotes a constant independent of parameters.

Prior Distribution of the Parameters: In USM, the two matrices $\mathbf{G} \in (0,1)^{d\times n}$ and $\mathbf{H} \in (0,1)^{d\times m}$ follow truncated Gaussian distributions:

$$p(\mathbf{G}|\sigma_g) = \prod_{i=1}^{n} \frac{\mathcal{N}(\mathbf{G}_i|0,\sigma_g^2\mathbf{I})}{\int_0^1 \mathcal{N}(x|0,\sigma_g^2)dx}, \tag{5.9}$$

$$p(\mathbf{H}|\sigma_h) = \prod_{j=1}^{m} \frac{\mathcal{N}(\mathbf{H}_j|0,\sigma_h^2\mathbf{I})}{\int_0^1 \mathcal{N}(x|0,\sigma_h^2)dx}, \tag{5.10}$$

where $\mathbf{G}_i$ and $\mathbf{H}_j$ are column vectors, and $\mathbf{I}$ is an identity matrix. Let denote $\varphi_g = \int_0^1 \mathcal{N}(x|0,\sigma_g^2)dx$ and $\varphi_h = \int_0^1 \mathcal{N}(x|0,\sigma_h^2)dx$. And $\boldsymbol{\zeta}^r \in \mathbb{R}^{n\times L}$ and $\boldsymbol{\zeta}^c \in \mathbb{R}^{m\times L}$ follow zero-mean spherical Gaussian:

$$p(\boldsymbol{\zeta}^r|\sigma_r^2) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{\zeta}_i^r|0,\sigma_r^2\mathbf{I}), \tag{5.11}$$

$$p(\boldsymbol{\zeta}^c|\sigma_c^2) = \prod_{j=1}^{m} \mathcal{N}(\boldsymbol{\zeta}_j^c|0,\sigma_c^2\mathbf{I}), \tag{5.12}$$

where $\boldsymbol{\zeta}_i^r$ and $\boldsymbol{\zeta}_j^c$ are column vectors.

In CRM, $\mathbf{U}$ and $\mathbf{V}$ follow zero-mean spherical Gaussian distribution:

$$p(\mathbf{U}|\sigma_u^2) = \prod_{i=1}^{n} \mathcal{N}(\mathbf{U}_i|\mathbf{0},\sigma_u^2\mathbf{I}), \tag{5.13}$$

$$p(\mathbf{V}|\sigma_v^2) = \prod_{j=1}^{m} \mathcal{N}(\mathbf{V}_j|\mathbf{0}, \sigma_v^2\mathbf{I}), \tag{5.14}$$

where $\mathbf{U}_i$ and $\mathbf{V}_j$ are column vectors.

5.4 Optimization

We use Expectation-Maximization (EM) [47], for convenience, to find the maximum a posterior estimates of the parameters of TPMF.

5.4.1 E-step

Both the rating matrix $\mathbf{R}$ and the item observability matrix $\mathbf{O}$ are partly available, we thus calculate the expectation of the ratings and item observability for missing entries, i.e., the entries with $s_{ij} = 0$. Note that we put rating expectation in the M-step via marginalizing $\mathbf{R}_{\mathscr{A}}$ for conveniently updating the latent factors.

Since the estimated rating values (i.e., $\hat{r}_{ij}$) for missing entries are continuous, we adopt a step function to scatter the values to $\{1,2,\cdots,L\}$ for the calculation of Equation (5.4). For simplicity, we partition $\mathbb{R}$ into L contiguous intervals with boundaries $b_0, b_1, \cdots, b_L$ where $b_0 = -\infty, b_1 = 1, \cdots, b_{L-1} = L-1, b_L = \infty$. r_{ij} is obtained according to the interval which the estimated rating belongs to: for example $r_{ij} = l$, if $b_{l-1} < \hat{r}_{ij} \leq b_l$. Since r_{ij} follows $\mathcal{N}(\hat{r}_{ij}, \sigma)$, we define:

$$p(r_{ij} = l|\hat{r}_{ij}) = \Phi\left(\frac{b_l - \hat{r}_{ij}}{\sigma}\right) - \Phi\left(\frac{b_{l-1} - \hat{r}_{ij}}{\sigma}\right), \tag{5.15}$$

where we denote $\phi(i,j,l) = p(r_{ij} = l|\hat{r}_{ij})$, and Φ is the cumulative distribution function for the standard Gaussian distribution:

$$\Phi(z) = Pr(\mathcal{N}(0,1) \leq z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2} dt. \tag{5.16}$$

Then, we obtain the expectation of o_{ij} below:

$$\begin{aligned} &\mathbb{E}(o_{ij}|\hat{r}_{ij}, \hat{s}_{ij}, \mu_{ij}, s_{ij} = 0) \\ &= \frac{\mu_{ij} \sum_{l=1}^{L} \phi(i,j,l) \mathcal{N}(0|\hat{s}_{ij}, \sigma_s^2)}{\mu_{ij} \sum_{l=1}^{L} \phi(i,j,l) \mathcal{N}(0|\hat{s}_{ij}, \sigma_s^2) + (1 - \mu_{ij})}. \end{aligned} \tag{5.17}$$

5.4.2 M-step

With respect to μ_j following the Beta distribution, we update μ_{ij} by finding the mode of the complete conditional $Beta(\alpha+\sum_i o_{ij}, \beta+n-\sum_i o_{ij})$ as below:

$$\mu_{ij} \leftarrow \frac{\alpha+\sum_i o_{ij}-1}{\alpha+\beta+n-2}. \tag{5.18}$$

To update the latent factors, we calculate the posterior probability given the estimated $\mathbf{O}$, i.e., $p(\mathbf{R}_{\mathscr{A}}, \mathbf{S}, \Omega|\mathbf{O}, \Theta)$, and separate the data into the available and missing parts to marginalize $\mathbf{R}_{\bar{\mathscr{A}}}$. Accordingly, we can obtain the posterior probability given the estimated $\mathbf{O}$ in E-step by separating the data into available and missing part:

$$\begin{aligned}
&p(\mathbf{R}_{\mathscr{A}}, \mathbf{S}, \Omega|\mathbf{O}, \Theta) \\
=&p(\mathbf{R}_{\mathscr{A}}|\Omega_r)p(\mathbf{S}_{\mathscr{A}}|\mathbf{R}_{\mathscr{A}}, \mathbf{O}_{\mathscr{A}}, \Omega_s)p(\Omega_r, \Omega_s|\Theta) \\
&\int_{\mathbf{R}_{\bar{\mathscr{A}}}} p(\mathbf{R}_{\bar{\mathscr{A}}}|\Omega_r)p(\mathbf{S}_{\bar{\mathscr{A}}}|\mathbf{R}_{\bar{\mathscr{A}}}, \mathbf{O}_{\bar{\mathscr{A}}}, \Omega_s)d\mathbf{R}_{\bar{\mathscr{A}}} \\
=&\prod_{(i,j)\in\mathscr{A}} \mathscr{N}(r_{ij}|\hat{r}_{ij}, \sigma)\left[\mathscr{N}(s_{ij}|\hat{s}_{ij}, \sigma_s^2)^{o_{ij}}\rho_0^{1-o_{ij}}\right] \\
&\prod_{(i,j)\in\bar{\mathscr{A}}} \left[\sum_{l=1}^{L}\phi(i,j,l)\mathscr{N}(s_{ij}|\hat{s}_{ij}, \sigma_s^2)\right]^{o_{ij}}\rho_0^{1-o_{ij}} \\
&\prod_{i=1}^{n}\frac{1}{\varphi_g}\mathscr{N}(\mathbf{G}_i|0, \sigma_g^2\mathbf{I})\mathscr{N}(\boldsymbol{\zeta}_i^r|0, \sigma_r^2\mathbf{I})\mathscr{N}(\mathbf{U}_i|\mathbf{0}, \sigma_u^2\mathbf{I}) \\
&\prod_{j=1}^{m}\frac{1}{\varphi_g}\mathscr{N}(\mathbf{H}_i|0, \sigma_h^2\mathbf{I})\mathscr{N}(\boldsymbol{\zeta}_i^c|0, \sigma_c^2\mathbf{I})\mathscr{N}(\mathbf{V}_i|\mathbf{0}, \sigma_v^2\mathbf{I}),
\end{aligned} \tag{5.19}$$

where we have $(i,j)\in\mathscr{A} \rightarrow s_{ij}=1, o_{ij}=1$ and $(i,j)\in\bar{\mathscr{A}} \rightarrow s_{ij}=0$. Then we obtain the log-likelihood of the posterior probability is given as below:

$$\begin{aligned}
&\log p(\mathbf{R}_{\mathscr{A}}, \mathbf{S}, \Omega|\mathbf{O}, \Theta) \\
=&\sum_{(i,j)\in\mathscr{A}} -\frac{1}{2\sigma^2}(\hat{r}_{ij}-r_{ij})^2 - \frac{1}{2\sigma_s^2}(\hat{s}_{ij}-1)^2 \\
+&\sum_{(i,j)\in\bar{\mathscr{A}}} o_{ij}\log(\mathscr{C}_{\bar{\mathscr{A}}_{ij}}) - \frac{||\mathbf{U}||_F}{2\sigma_u^2} - \frac{||\mathbf{V}||_F}{2\sigma_v^2} - \frac{||\mathbf{G}||_F}{2\sigma_g^2} \\
-&\frac{||\mathbf{H}||_F}{2\sigma_h^2} - \frac{1}{2\sigma_r^2}||\boldsymbol{\zeta}^r||_F - \frac{1}{2\sigma_c^2}||\boldsymbol{\zeta}^c||_F + \mathscr{C},
\end{aligned} \tag{5.20}$$

where $\mathscr{C}_{\bar{\mathscr{A}}_{ij}} = \sum_{l=1}^{L}\phi(i,j,l)\mathscr{N}(0|\hat{s}_{ij}, \sigma_s^2)$ and $\mathscr{C}$ is a constant independent of parameters. Since $\sum_{l=1}^{L}\phi(i,j,l)=1$ and the logarithm function (log) is concave, based on the property

of concave functions, we have:

$$\log \sum_{l=1}^{L} \phi(i,j,l) \mathcal{N}(0|\hat{s}_{ij}, \sigma_s^2) \geq \sum_{l=1}^{L} \phi(i,j,l) \log \mathcal{N}(0|\hat{s}_{ij}, \sigma_s^2) \tag{5.21}$$

Therefore, our objective to maximize $\log p(\mathbf{R}_{\mathscr{A}}, \mathbf{S}, \Omega | \mathbf{O}, \Theta)$ is equal to maximize its infimum:

$$\begin{aligned} \mathscr{L}(\Omega, \Theta) = & \sum_{(i,j) \in \mathscr{A}} -\frac{1}{2\sigma^2}(\hat{r}_{ij} - r_{ij})^2 - \frac{1}{2\sigma_s^2}(\hat{s}_{ij} - 1)^2 \\ & + \sum_{(i,j) \in \bar{\mathscr{A}}} o_{ij} \sum_{l=1}^{L} \phi(i,j,l) (\frac{\hat{s}_{ij}^2}{2\sigma_s^2} + \rho) - \frac{||\mathbf{U}||_F}{2\sigma_u^2} - \frac{||\mathbf{V}||_F}{2\sigma_v^2} \\ & - \frac{||\mathbf{G}||_F}{2\sigma_g^2} - \frac{||\mathbf{H}||_F}{2\sigma_h^2} - \frac{1}{2\sigma_r^2}||\boldsymbol{\zeta}^r||_F - \frac{1}{2\sigma_c^2}||\boldsymbol{\zeta}^c||_F + \mathscr{C}. \end{aligned} \tag{5.22}$$

where $\rho = \log \frac{1}{\sqrt{2\pi\sigma_s^2}}$ is independent to Ω.

Then, we calculate the log-likelihood of the probability and obtain the objective function.

$$\begin{aligned} \mathscr{L}(\Omega, \Theta) = & \sum_{(i,j) \in \mathscr{A}} -\frac{1}{2\sigma^2}(\hat{r}_{ij} - r_{ij})^2 - \frac{1}{2\sigma_s^2}(\hat{s}_{ij} - 1)^2 \\ & + \sum_{(i,j) \in \bar{\mathscr{A}}} o_{ij} \sum_{l=1}^{L} \phi(i,j,l) (\frac{\hat{s}_{ij}^2}{2\sigma_s^2} + \rho) - \frac{||\mathbf{U}||_F}{2\sigma_u^2} - \frac{||\mathbf{V}||_F}{2\sigma_v^2} \\ & - \frac{||\mathbf{G}||_F}{2\sigma_g^2} - \frac{||\mathbf{H}||_F}{2\sigma_h^2} - \frac{||\boldsymbol{\zeta}^{row}||_F}{2\sigma_r^2} - \frac{||\boldsymbol{\zeta}^{col}||_F}{2\sigma_c^2} + \mathscr{C}. \end{aligned} \tag{5.23}$$

where $\rho = \log \frac{1}{\sqrt{2\pi\sigma_s^2}}$ is independent of Ω and $\mathscr{C}$ is a constant. Our objective is to maximize $\mathscr{L}(\Omega, \Theta)$ to learn an optimal of $\Omega = \{\mathbf{U}, \mathbf{V}, \mathbf{G}, \mathbf{H}, \boldsymbol{\zeta}^{row}, \boldsymbol{\zeta}^{col}, b_s\}$ under the hyperparameter Θ. Since $\mathscr{L}(\Omega, \Theta)$ has no analytical solution, we take batch gradient ascent to update Ω following Yang et al. [258].

The resulting optimization algorithm shown in Algorithm 1 belongs to the class of *generalized EM algorithms* guaranteed to converge to a (local) optimum of the log-likelihood [63, 241].

5.4.3 Gradients of the Parameters.

To calculate the gradients, we provide the following denotation for convenience:

$$\delta(i,j,l) = \frac{exp(-\frac{(b_l - \hat{r}_{ij})^2}{2\sigma^2}) - exp(-\frac{(b_l - \hat{r}_{ij})^2}{2\sigma^2})}{\sqrt{2\pi}\sigma}, \tag{5.24}$$

Algorithm 1 Generalized EM for TPMF

1: **Input:** Rating dataset $\mathscr{D}$
2: Obtain triple aspects of $\mathscr{D}$: rating matrix $\mathbf{R}$, item observability matrix $\mathbf{O}$, and user selection matrix $\mathbf{S}$
3: Initialize $\Omega = \{\mathbf{U}, \mathbf{V}, \mathbf{G}, \mathbf{H}, \boldsymbol{\zeta}^{row}, \boldsymbol{\zeta}^{col}, b_s\}$, $\boldsymbol{\mu}$
4: **while** *stopping criteria is not satisfied* **do**
5: Compute the expected item observability for missing entries, i.e., $\mathbf{O}_{\bar{\mathscr{A}}}$, by Equation (5.17)
6: Update Ω by batch gradient ascent along the gradient $\nabla_{\Omega}\mathscr{L}(\Omega, \Theta)$
7: Update $\boldsymbol{\mu}$ by Equation (5.18)
8: **end while**

$$\boldsymbol{\Lambda}_{ij} = \frac{diag(\mathbf{U}_i \circ \mathbf{V}_j)d}{\mathbf{G}_i^T \mathbf{H}_j + d\varepsilon}, \tag{5.25}$$

$$s_{ij}^{\rho} = \frac{\hat{s}_{ij}^2}{2\sigma_s^2} + \rho. \tag{5.26}$$

Then, the gradients for parameters $\mathbf{U}, \mathbf{V}, \mathbf{G}, \mathbf{H}, \boldsymbol{\zeta}^r, \boldsymbol{\zeta}^c, b_s$ are listed below:

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \mathbf{U}_i} = & \sum_{j,(i,j)\in\mathscr{A}} -\frac{1}{\sigma^2}(\mathbf{U}_i^T \boldsymbol{\Gamma}_{ij} \mathbf{V}_j - r_{ij})\boldsymbol{\Gamma}_{ij}\mathbf{V}_j \\ & - \sum_{j,(i,j)\in\bar{\mathscr{A}}} o_{ij} \sum_{l=1}^{L} \delta(i,j,l) s_{ij}^{\rho} \boldsymbol{\Gamma}_{ij}\mathbf{V}_j - \frac{1}{\sigma_u^2}\mathbf{U}_i \end{aligned} \tag{5.27}$$

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \mathbf{V}_j} = & \sum_{i,(i,j)\in\mathscr{A}} -\frac{1}{\sigma^2}(\mathbf{U}_i^T \boldsymbol{\Gamma}_{ij} \mathbf{V}_j - r_{ij})\boldsymbol{\Gamma}_{ij}^T\mathbf{U}_i \\ & - \sum_{i,(i,j)\in\bar{\mathscr{A}}} o_{ij} \sum_{l=1}^{L} \delta(i,j,l) s_{ij}^{\rho} \boldsymbol{\Gamma}_{ij}^T\mathbf{U}_i - \frac{1}{\sigma_u^2}\mathbf{V}_j \end{aligned} \tag{5.28}$$

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \mathbf{G}_i} = & \sum_{j,(i,j)\in\mathbf{A}} -\frac{1}{\sigma^2}(\mathbf{U}_i^T \boldsymbol{\Gamma}_{ij} \mathbf{V}_j - r_{ij})\boldsymbol{\Lambda}_{ij}\mathbf{H}_j - \frac{1}{\sigma_s^2}(\hat{s}_{ij} - 1)\mathbf{H}_j \\ & - \sum_{j,(i,j)\in\bar{\mathscr{A}}} o_{ij} \sum_{l=1}^{L} \delta(i,j,l) s_{ij}^{\rho} \Lambda_{ij}\mathbf{H}_j + \phi(i,j,l)\frac{\hat{s}_{ij}}{\sigma_s^2}\mathbf{H}_j - \frac{1}{\sigma_g^2}\mathbf{G}_i \end{aligned} \tag{5.29}$$

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \mathbf{H}_j} = & \sum_{i,(i,j)\in\mathbf{A}} -\frac{1}{\sigma^2}(\mathbf{U}_i^T \boldsymbol{\Gamma}_{ij} \mathbf{V}_j - r_{ij})\boldsymbol{\Lambda}_{ij}^T\mathbf{G}_i - \frac{1}{\sigma_s^2}(\hat{s}_{ij} - 1)\mathbf{G}_i \\ & - \sum_{i,(i,j)\in\bar{\mathscr{A}}} o_{ij} \sum_{l=1}^{L} \delta(i,j,l) s_{ij}^{\rho} \Lambda_{ij}^T\mathbf{G}_i + \phi(i,j,l)\frac{\hat{s}_{ij}}{\sigma_s^2}\mathbf{G}_i - \frac{1}{\sigma_h^2}\mathbf{H}_j \end{aligned} \tag{5.30}$$

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial \boldsymbol{\zeta}^r_{il}} = &\sum_{j,(i,j)\in\mathscr{A}} \frac{1}{\sigma_s^2}\mathbb{I}(r_{ij}=l)(\hat{s}_{ij}-1) \\ &- \sum_{j,(i,j)\in\bar{\mathscr{A}}} o_{ij}\phi(i,j,l)\frac{\hat{s}_{ij}}{\sigma_s^2} - \frac{1}{\sigma_r^2}\boldsymbol{\zeta}^r_{il}\end{aligned} \tag{5.31}$$

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial \boldsymbol{\zeta}^c_{jl}} = &\sum_{i,(i,j)\in\mathscr{A}} \frac{1}{\sigma_s^2}\mathbb{I}(r_{ij}=l)(\hat{s}_{ij}-1) \\ &- \sum_{i,(i,j)\in\bar{\mathscr{A}}} o_{ij}\phi(i,j,l)\frac{\hat{s}_{ij}}{\sigma_s^2} - \frac{1}{\sigma_c^2}\boldsymbol{\zeta}^c_{jl}\end{aligned} \tag{5.32}$$

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial b_c} = &\sum_{(i,j)\in\mathscr{A}} \frac{1}{\sigma_s^2}(r_{ij}=l)(\hat{s}_{ij}-1) \\ &- \sum_{(i,j)\in\bar{\mathscr{A}}} o_{ij}\sum_{l=1}^{L}\phi(i,j,l)\frac{\hat{s}_{ij}}{\sigma_s^2}\end{aligned} \tag{5.33}$$

5.4.4 Discussion

Let us calculate the likelihood probability of an entry being missing (unselected), i.e., $s_{ij}=0$ by marginalizing r_{ij} and o_{ij}:

$$\begin{aligned}&p(s_{ij}=0|\mu_{ij},\Omega,\Theta) \\ =&\int_{o_{ij}}\int_{r_{ij}} p(s_{ij}=0,o_{ij},r_{ij}|\mu_{ij},\Omega,\Theta)dr_{ij}do_{ij} \\ =&(1-\mu_{ij})+\mu_{ij}\sum_{l=1}^{L}\phi(i,j,l).\mathscr{N}(0|\hat{s}_{ij},\sigma_s^2)\end{aligned} \tag{5.34}$$

with regarding to maximizing the likelihood, we find that μ_{ij} downweights the probability of the missing entries being negative (i.e., $\hat{s}_{ij}=0$), and the smaller μ_{ij} corresponds to the higher probability of $\hat{s}$ not being 0.

Since the missing entries are partly attributed to the other entries being unobservable (i.e., $o_{ij}=0$), when we set $\mu_{ij}=1$ for all entries, the TPMF model degrades to the classic MNAR models (e.g., Logitvd and MF-MNAR) which treat all missing entries as negative ones, which is intuitively not the real case.

5.5 Experiments

Since it is difficult to obtain unavailable ratings for testing, we first generate synthetic data to mimic different types of MNAR data and conduct experiments to investigate

the effectiveness and robustness of TPMF in handling MNAR ratings. We then compare TPMF against several state-of-the-art methods on four real-world datasets.

5.5.1 Datasets

To investigate the effective of TPMF in handling MNAR data and explore how different observation and selection schemes affect the rating prediction of TPMF, we construct six synthetic datasets to experimentally control the selection biases and also evaluate the performance of TPMF on four real-world rating datasets. **Synthetic Datasets.** The synthetic datasets are generated by a matrix factorization model. First, we set $n = m = 1,000$, $d = 10$ and $L = 5$ and generate the matrices $\mathbf{U}$, $\mathbf{V}$, $\boldsymbol{\zeta}^r$ and $\boldsymbol{\zeta}^c$ from the standard Gaussian and $\mathbf{G}$ and $\mathbf{H}$ from a uniform distribution within $[0,1]$. Then, we generate the integer ratings by $r_{ij} = \lceil L \times \psi(\mathbf{U}_i \boldsymbol{\Gamma}_{ij} \mathbf{V}_j^T) \rceil$ and draw o_{ij} from $Bernoulli(\mu_j)$ where μ_j is drawn from $Beta(2, \beta_0)$. Accordingly, we assign the selection variable $s_{ij} = 1$ with a probability of $\rho_s \delta\left(\mathbf{G}_i \mathbf{H}_j^T + \sum_{l=1}^{L} z_l \mathbf{I}[r_{ij} = l] - 2\right)/Z$ when $o_{ij} = 1$, and $s_{ij} = 0$ otherwise. Here, δ is a logistic function and $(z_1, \cdots, z_5) = (-2, -2, -2, 2, 2)$ reflects items with high ratings are more likely to be selected and Z is used to normalize the probability. The ratings with $s_{ij} = 1$ are selected to construct the dataset. Here, β_0 and ρ_s are used to control the global observability (i.e., #observable items per nm, denoted $p_o \in (0,1]$) and rating density (i.e., #ratings per nm, denoted $d_r \in (0,1)$). Roughly, we have $\beta_0 = 2/p_o - 2$ and $\rho_s = d_r/p_o$. We denote this synthetic data as **DDC** which indicates the combination of *item-dependent* observability scheme, *rating-dependent* selection scheme, and *factor-correlated* rating scheme.

To investigate how different observability, selection and rating schemes affect the prediction performance of TPMF, we change scheme combination based on DDC and generate another three datasets: 1) **RDC** - using *random* observability scheme, i.e., $o_{ij} \sim Bernoulli(p_o)$; 2) **DDU** - changed to *factor-uncorrelated* rating scheme, i.e., $r_{ij} = \lceil L \times \psi(\mathbf{U}_i \mathbf{V}_j^T) \rceil$; 3) **DRU** - using *random* selection scheme, i.e., $s_{ij} | o_{ij} = 1 \sim Bernoulli(d_r/p_o)$, and the *factor-uncorrelated* rating scheme; and 4) **RRU** - using *random* observability and selection schemes and *factor-uncorrelated* rating scheme. During the generation, we tune β_0 and p_s to keep the global observability p_o and rating density d_r nearly the same. For all synthetic datasets, we randomly sample two test sets: a *standard set* sampled from the available ratings r_{ij} with $s_{ij} = 1$ and a *special set* sampled from the missing rating r_{ij} with $s_{ij} = 0$, and treat the rest of the available ratings as the training set. For all synthetic datasets, we tune β_0 and p_s to keep the global observability p_o and rating density d_r nearly the same. Meanwhile, we randomly select 10% missing ratings as the

MAR testing data for each combination.

Real-world Datasets. The evaluation of debiasing rating estimation should be verified on MAR ratings. four real-world rating datasets with MAR ratings are considered: 1) Yahoo R3 (denoted Yahoo)[1] collects 311,704 MNAR ratings and 45,000 MAR ratings from 15,400 users on 1,000 songs. 2) The Coat (Coat)[2] has 6,960 MNAR ratings and 4,640 ratings of 290 users to 300 coats. And we collect another two real-world datasets that only have MNAR ratings: 3) MovieLens-1M (ML1M)[3] contains about 1M MNAR ratings from 6,040 users and 3,706 movies. 4) The Movie Tweetings (MTweet)[4] collects 106,337 MNAR ratings by 3,972 users on 2,043 movies from Twitter, where we rescale the original ratings from [0; 10] to the interval [1; 5]. We use MNAR ratings for training and MAR ratings for testing on Yahoo and Coat, while we randomly split the dataset into training/test sets with 80/20 proportions on ML1M and MTWeet. Since there are no MAR ratings in ML1M and MTWeet, we set aside 5% of the MNAR ratings and use Naive Bayes to learn propensities.

5.5.2 Experimental Settings

Baseline Methods. We compare TPMF with one basic approach and four state-of-the-art debiasing approaches, including: 1) **PMF** [179] which is based on MAR assumption; 2) **MF-MNAR** [77] which deals with the MNAR nature of rating data based on jointly learning the missing data model and the complete rating model; 3) **MF-IPS** [185] which develops an unbiased estimator for the MNAR rating data based on the Inverse-Propensity-Scoring (IPS); 4) **MF-JL**; and 5) **MF-DR-JL** [233] which propose a more robust unbiased estimator by integrating IPS and estimated imputed errors for the MNAR rating data. Besides, we introduce two variants of the proposed model: **T-FO** treating all items being fully observed, i.e., $o_{ij} = 1$, and **T-NF** neglecting the factor correlation between ratings and selection, i.e., prediction ratings by $\hat{r}_{ij} = \mathbf{U}_i^T \mathbf{V}_j$.

Parameter Settings. We utilize the mean absolute error (MAE) and root mean squared error (RMSE) to evaluate the experimental results. For a fair comparison, we tune the hyperparameters on validation sets by grid search and obtain the best for testing. Specifically, we choose the latent dimension d in $\{10, 20, 30, 40\}$, learning rate in $\{0.01, 0.05, 0.1, 1\}$, and L_2 regularization rate in $\{0.01, 0.1, 1\}$ (if required) and keep

[1] https://webscope.sandbox.yahoo.com/
[2] https://www.cs.cornell.edu/ schnabts/mnar/
[3] https://grouplens.org/datasets/movielens/
[4] http://github.com/sidooms/MovieTweetings

Table 5.1: Performance of TPMF compared against PMF and its variants on the five synthetic datasets ($p_o = 0.5$ and $d_r = 0.1$).

Dataset	Metric	Special Test Set				Standard Test Set			
		PMF	T-FO	T-NF	TPMF	PMF	T-FO	T-NF	TPMF
RRU	MAE	0.2779	0.2677	**0.2651**	0.2696	0.2792	0.2685	**0.2651**	0.2707
	RMSE	0.3357	0.32	**0.3158**	0.3229	0.3359	0.3207	**0.3157**	0.3239
DRU	MAE	0.2758	0.2667	**0.2628**	0.2649	0.2856	0.2723	**0.2664**	0.2703
	RMSE	0.33	0.3169	**0.3114**	0.3144	0.349	0.3271	**0.3184**	0.3243
DDU	MAE	0.2765	0.2614	0.2598	**0.2573**	0.2897	0.2773	0.2769	**0.2758**
	RMSE	0.3325	0.311	0.3102	**0.3088**	0.3504	0.3326	0.3324	**0.3299**
RDC	MAE	0.2873	0.2718	0.273	**0.2699**	0.289	0.2737	0.2745	**0.2723**
	RMSE	0.3461	0.3251	0.3254	**0.3241**	0.3527	0.3307	0.3319	**0.3281**
DDC	MAE	0.2901	0.2751	0.2742	**0.2719**	0.2949	0.2828	0.2822	**0.2821**
	RMSE	0.3498	0.3294	0.3288	**0.3274**	0.3591	0.3421	0.3414	**0.3404**

other hyperparameters recommended from the source codes of the baselines. Regarding TPMF, we fix $\alpha = \beta = 1$ and $\varepsilon = 0.5$ for simplicity. Beside, we tune $\lambda_s = \sigma^2/\sigma_s^2$, $\lambda_v = \sigma^2/\sigma_v^2$, $\lambda_u = \sigma^2/\sigma_u^2$, $\lambda_g = \sigma^2/\sigma_g^2$, $\lambda_h = \sigma^2/\sigma_h^2$, $\lambda_r = \sigma^2/\sigma_r^2$ and $\lambda_c = \sigma^2/\sigma_c^2$ over $\{0.01, 0.1, 1, 10\}$, the learning rate over $\{0.005, 0.01, 0.05, 0.1\}$, and L_2 regularization rate over $\{0.1, 0.5, 1\}$. To guarantee a fast convergence and avoid overfitting, we further initialize $\mathbf{U}$ and $\mathbf{V}$ from a pretrained PMF model and initialize $\boldsymbol{\mu}$ with item frequency.

5.5.3 Experimental Results

Synthetic Experiments. To analyze the effectiveness of TPMF, we evaluate TPMF and its two variants on the five synthetic datasets. Results reporting MAE and RMSE on the special test data and standard test data are provided in Table 5.1. The results show that the proposed TPMF and its variants outperform the biased method PMF. Considering the characteristics of the datasets, we notice that TPMF performs the best under both metrics except on RRU and DRU. This is reasonable because: 1) TPMF models item observability and factor correlation to handle both simple (i.e., RDC) and complex (i.e., DDC and DDU) item observability schemes and are suitable for the cases with existence (i.e., DDC and RDC) and nonexistence (i.e., DDU) of factor correlation. 2) Relative to the other datasets, both DRU and RRU are simple and randomly select ratings without adding factor correlation. In this case, TPMF may overfit these two datasets and degrade the prediction performance. Comparing the two tables, we see that

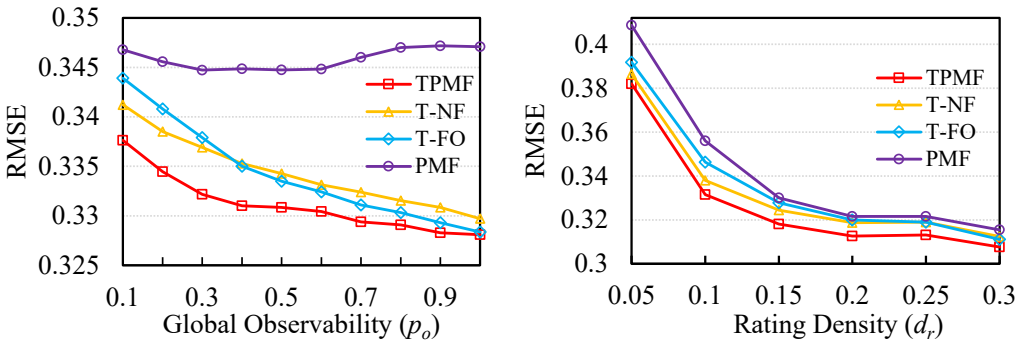


Figure 5.3: Evaluation on dataset DDC with varying global observability and rating density.

all models perform better on standard test data than on special test data except on RRU which is an MAR dataset, confirming that the MNAR issues degrade the generalization of the model trained on the biased data to random data. Overall, the results indicate that TPMF can effectively model item observability, user selection and ratings, and infer the relationships between the triple aspects.

Robustness Study. We further investigate the performance of the proposed methods on DDC with varying global observability rates (i.e., $p_o \in \{0.1, 0.2, \cdots, 1.0\}$) and rating density levels (i.e., $den \in \{0.05, 0.1, \cdots, 0.25, 0.03\}$). Results reporting RMSE on special test sets are provided in Figure 5.3, where we observe that the proposed methods achieve higher prediction accuracy than PMF. In terms of item observability, higher p_o improves higher prediction accuracy for the proposed methods, which is attributed to the fact that higher item observability simplifies the dataset (note that PMF is not sensitive to the simplicity) and benefits the inference of the two sub-models USM and CRM (see the discussion in Inference). And T-FO performs worse than T-NF when the global observability p_o is small and catches up and exceeds T-NF when $p_o > 0.4$, which confirms that capturing factor correlation plays more important roles with item observability increasing. In addition, all methods obtain obvious improvement with the increase of rating density, which is attributable since more ratings intuitively facilitate the inference of rating generation.

Performance Comparison. To further investigate the effectiveness of TPMF, we report the performance of TPMF and baseline methods on real-world datasets in Table 5.2, where the best performance on each dataset is marked in bold. Imp reports the performance improvement of TPMF over the best baseline. Our TPMF outperforms the state-of-art methods under both metrics on all datasets. Note that MF-IPS performs

Table 5.2: Performance of TPMF compared against its variants and the state-of-the-art baselines on four real-world datasets.

Dataset	Metric	PMF	MF-MNAR	MF-IPS	MF-JL	MF-DR-JL	T-FO	T-NF	TPFM	$Imp.(\%)$
Coat	MAE	0.736	0.704	0.735	0.69	0.701	0.697	0.679	**0.67**	2.99
	RMSE	0.934	0.899	0.927	0.883	0.897	0.893	0.869	**0.857**	3.03
Yahoo	MAE	0.973	0.956	0.918	0.903	0.804	0.907	0.821	**0.771**	4.28
	RMSE	1.223	1.196	1.215	1.182	1.177	1.186	1.172	**1.165**	3.23
ML1M	MAE	0.701	0.691	0.702	0.671	0.68	0.684	0.671	**0.662**	1.36
	RMSE	0.886	0.878	0.89	0.857	0.865	0.869	0.856	**0.845**	1.42
MTWeet	MAE	0.556	0.519	0.53	0.511	0.502	0.521	**0.492**	**0.493**	3.65
	RMSE	0.741	0.692	0.701	0.685	0.661	0.695	**0.651**	**0.652**	5.22

worse than other debiasing methods and even PMF on the MovieLens dataset while MF-MNAR, MF-JL and MF-DR-JL achieve desirable performance on all the datasets. The results are well explainable. IPS-based methods debias rating estimates by inducing the knowledge of the selection bias and guarantee no bias (if the propensities are correct) but high variance. Meanwhile, the imputation-base methods, i.e., MF-MNAR, rely on modeling the entire generation process of rating to counterfactually estimate ratings, which gives non-zero bias but very low/zero variance. MF-JL and MF-DR-JL get the best of both the worlds i.e. no bias when either of the models is unbiased and lower variance than IPS. Hence, one might expect that a method like MF-DR-JL using TPMF instead of the MF-MNAR would lead to better results.

In addition, TPMF shows clear advantages over the comparative methods on Coat and Yahoo (two MAR test sets) relative to its performance on ML1M and MTWeet. Deep insight behind the superior results lays that jointly considering item observability, user selection and ratings facilitates debiasing the rating estimation on MNAR data, and, more importantly, TPMF effectively models the triple aspects. Table 5.2 also reports T-NF performs better than T-FO, indicating that item observability plays a more important role than factor correlation in debiasing rating estimation. This may be caused by the fact that a large number of items are unobservable to users in practical recommendation data.

5.6 Conclusions

In this chapter, we aim to address the rating distribution heterogeneity issues to debias rating estimate. Specifically, we model the rating generation process with a tripartite collaborative filtering model for jointly inferring triple aspects: item observability, user

selection and ratings.

To the best of our knowledge, this work represents the first attempt to address the missing-not-at-random ratings by exploring the complex dependencies between item observability, user selection, and ratings. Extensive empirical results show that modeling item observability and user selection is essential and can effectively debias rating estimation in the MNAR data, and our model outperforms the existing state-of-the-art methods for the MNAR data.

Part III

Non-IID MTS Analysis

CHAPTER 6

SPECTRAL CLUSTERING-ENHANCED TRANSFORMER FOR NON-IID MULTIVARIATE TIME SERIES

6.1 Introduction

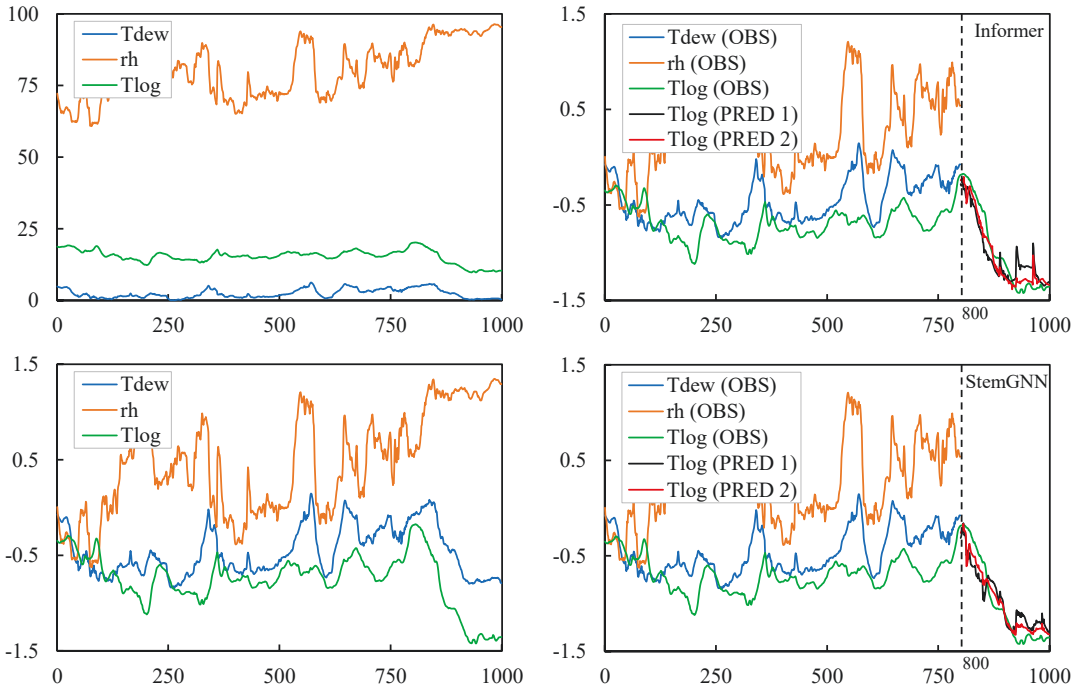
Modeling non-IID MTS is challenging, requiring inter-TS couplings to be characterized with heterogeneous variable scales and distributions [21, 293]. Existing models often neutralize heterogeneity by explicitly (in observations) or implicitly (in hidden feature space) modeling MTS under the IID assumption. They normalize heterogeneous MTS and transform MTS inputs to a similar scale, which are then fed into subsequent DNNs such as long short-term memory (LSTM) [111] and convolutional neural networks (CNNs) [37, 289]. The normalization neutralizing different variable scales alleviates the influence of dominant time series variables and facilitates fast model training and convergence. However, it neglects the impact of heterogeneous variable scales and distributions on inter-TS couplings in scale and distribution-varying MTS, leading to possibly inferior MTS modeling. As illustrated in Figure 6.1(a), the raw TS are deformed after normalization. The deformation during normalization distorts the inter-TS couplings in scale and distribution, which conceals the inter-TS heterogeneity in the subsequent modeling of inter-TS relations.

In addition, not all MTS may be relevant to forecasting target time series. For example, some may be independent, redundant, or noisy. In Figure 6.1(b), the forecasting performance of Informer [289] and StemGNN [18] on two related TS *Tdew* and *Tlog* is

better than those based on all three TS *Tdew*, *rh* and *Tlog*. Intuitively, irrelevant time series variables may not contribute to or even harm the forecasting on each other. This further inspires us to differentiate target-irrelevant and -relevant TS during learning MTS couplings and heterogeneities for forecasting. However, most neural MTS models uniformly concatenate all time series variables at every time step to a one-strand vector and then feed the multi-step vectors into networks. They usually ignore or weaken the time series relevance/irrelevance and the aforementioned complex couplings and heterogeneities in modeling non-IID MTS. Despite capably downweighting non-informative features and obtaining effective features, they hardly learn optimal variable correlations without specific and explicit objectives for the non-IIDness and relevance across MTS.

Motivated by the aforementioned MTS non-IIDness and relevance, this work models non-IID MTS capturing heterogeneous time series couplings for MTS forecasting: a) effectively modeling inter-TS heterogeneities, and b) accurately segregating MTS to obtain target-relevant MTS for forecasting. The related work addresses variable heterogeneity by embedding time series variables into a unified space. Then, variable embeddings are multiplied with the variable values to represent the heterogeneous MTS input [171]. In addition, time series clustering emerges in learning inter-TS similarity [71, 115, 145] by separating dissimilar TS from similar ones. This effectively avoids irrelevant variables and facilitates variable heterogeneities in each cluster (similar to the divide-and-conquer approach). However, existing time series clustering methods rarely cooperate with MTS forecasting models. They cannot jointly characterize MTS couplings, heterogeneities, or target relevance. This inspires our efforts to leverage time series clustering in enhancing MTS forecasting.

Accordingly, a novel non-IID MTS forecaster Cospectrumer is proposed to effectively capture both target-relevant inter- and intra-TS couplings and variable heterogeneities in MTS. Cospectrumer integrates spectral clustering and Transformer. A spectral clustering network segregates MTS, and a clusterwise forecasting network performs MTS forecasting. The clustering network adopts discrete Fourier transform (DFT) to extract the frequency-domain features, a convolutional layer to downsample the frequency features, and fully connected layers to extract high-level features. This design theoretically guarantees the Lipschitz continuity [211] for better model generalization and interpretability. Then, the spectral relaxation of the K-means objective is utilized to learn a cluster indicator matrix and generate clusterwise inputs accordingly. On the other hand, the forecasting network adopts the encoder-decoder architecture and introduces k-channel Transformers corresponding to k clusters to model intra- and inter-TS cou-



(a) Data Analysis (b) Prediction Analysis

Figure 6.1: An illustration of the normalization and relevance effect on MTS forecasting on weather data.

It contains three time series variables: *Tdew*, *rh*, and *Tlog*. *Tdew* records the temperatures of dewpoints, *rh* records the relative humidity, and *Tlog* records the temperatures of loggers. *X-axis* and *Y-axis* denote time steps and scales, respectively. (a) Data analysis: the raw values and normalized values of the variables are visualized on the top and bottom respectively; (b) Prediction analysis: the observed values (*OBS*) of the previous 800 steps are utilized to predict the values of the successive 200 steps of variable *Tlog* by Informer (on the top) and StemGNN (at the bottom). We perform the predictions of *Tlog* based on the observations of all three variables and only the related variables *Tdew* and *Tlog* respectively, corresponding to *Tlog (PRED 1)* and *Tlog (PRED 2)*.

plings. The network includes a heterogeneous embedding module to represent the mixed inputs of variables, timestamps, and positional information and learn second-order TS couplings using the outer product. In Transformers, a cluster-aware multi-head attention mechanism guarantees information isolation between clusters and allocates the co-attention between different clusterwise inputs. Finally, we obtain clusterwise decoder outputs and feed them to fully connected prediction layers for clusterwise forecasting. In a closed form, Cospectrumer jointly learns the two modules for an effective time series clustering to improve MTS forecasting.

6.2 Problem Formulation

Let $\mathbf{x}_t \in \mathbb{R}^N$ be the multivariate variables of N dimension at time step t and $\mathbf{X}_T = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T\} \in \mathbb{R}^{N \times T}$ be the total observations with T time steps. Under the rolling forecasting settings, given a sequence of historical L-steps of observations on the variables at time step t, $\mathbf{X}_L^t = \{\mathbf{x}_{t-L+1}, \mathbf{x}_{t-L+2}, \cdots, \mathbf{x}_t\} \in \mathbb{R}^{N \times L}$, our goal is to predict the following D-step values on the variables, i.e., $\mathbf{Y}^t = \{\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \cdots, \mathbf{x}_{t+D}\} \in \mathbb{R}^{N \times D}$. To address this problem, Cospectrumer combines a spectral clustering network and a clusterwise forecasting network, as shown in Figure 6.2. We introduce those below.

6.3 The Cospectrumer Model

6.3.1 Spectral Clustering Network

We propose a novel and concise hybrid clustering method, a spectral clustering network shown in Figure 6.2, to segregate time series variables. On each univariate TS, DFT is first applied to obtain its frequency spectrum. A convolutional layer further obtains mixed spectral features, which also captures the correlations between different frequency bands. Given the raw time series $\mathbf{X}_T$, the corresponding process is formulated as follows:

$$\mathbf{F} = \text{CONV}(\mathscr{F}_{DFT}(\mathbf{X}_T)/T) \tag{6.1}$$

where $\mathscr{F}_{DFT}$ and CONV denote the operators of DFT and convolution respectively. For long MTS, the output spectrum of DFT grows large in magnitude, easily pushing subsequent activation functions into regions where it has extremely small gradients. To counteract this effect, we scale the output of DFT by $\frac{1}{T}$ with T being the time series length. In addition, $\mathbf{F} \in \mathbb{R}^{N \times L_d}$ denotes the mixed frequency spectral features, where we adopt the convolutional layer to downsample L_d frequency points from the output of DFT to avoid overlong features. Note that we perform DFT and convolution on each time series separately. We then feed the spectral features of N TS into ℓ fully connected layers to obtain high-level nonlinear features.

$$\mathbf{F}_\ell = \sigma_\ell((\cdots \sigma_1(\mathbf{F} \cdot \mathbf{W}_1 + \mathbf{b}_1)\cdots)\mathbf{W}_\ell + \mathbf{b}_\ell) \tag{6.2}$$

where $\{\sigma_i\}_{i=1}^{\ell}$, $\{\mathbf{W}_i\}_{i=1}^{\ell}$ and $\{\mathbf{b}_i\}_{i=1}^{\ell}$ denote the activation functions (precisely tanH used in Cospectrumer), weights, and biases of the multiple fully connected layers, respectively. To learn the time series clustering, we guide the clustering network using the K-means objective. Following the spectral relaxation conversion of the K-means objective [272], we

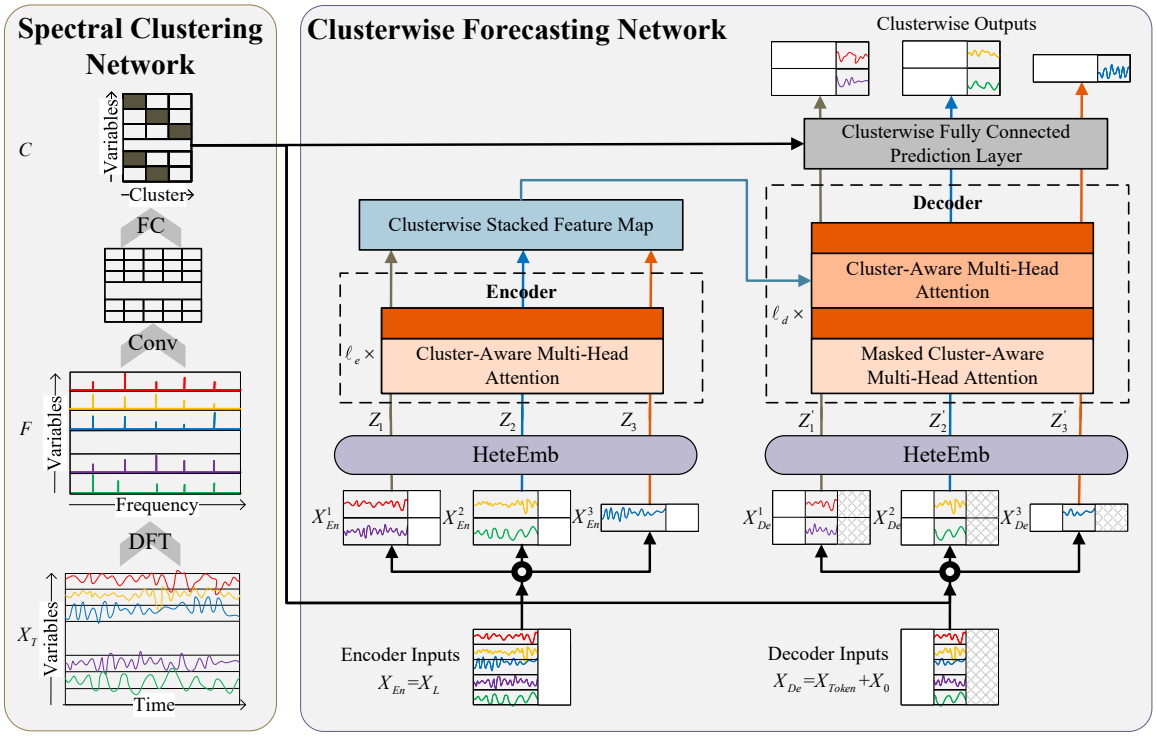


Figure 6.2: The architecture of Cospectrumer, illustrated with $k = 3$ in the K-means objective.

It comprises two components: a spectral clustering network, and a clusterwise forecasting network. FC and CONV denote fully connected layers and convolutional layers, respectively. The orange box denotes the pipeline of add&normalization&feed-forward layers. The binary cluster indicator matrix $\mathbf{C}$ is denoted by black (1) and blank (0) boxes. The blank rectangles denote the truncation of time series, while the shadowed rectangles denote masking the time series with placeholders.

formulate the K-means objective as a minimization problem associated with the Gram matrix $\mathbf{F}_\ell \mathbf{F}_\ell^T$ as follows:

$$\begin{aligned} \mathscr{L}_{KM} &= \min_{\mathbf{C},\mathbf{W},\mathbf{b}} (\mathrm{Tr}(\mathbf{F}_\ell \mathbf{F}_\ell^T) - \mathrm{Tr}(\mathbf{C}^T \mathbf{F}_\ell \mathbf{F}_\ell^T \mathbf{C})), \\ & s.t.\ \mathbf{C}^T \mathbf{C} = \mathbf{I}_k. \end{aligned} \tag{6.3}$$

where $\mathrm{Tr}(\cdot)$ denotes the track function, and $\mathbf{C} \in \{0,1\}^{N \times k}$ is the cluster indicator matrix of k clusters whose element $c_{ij} = 1$ denotes that the i-th time series belongs to the j-th cluster, otherwise not. $\mathbf{I}_k$ is a diagonal matrix of $\mathrm{diag}(\mathrm{s}_1, \cdots, \mathrm{s}_\mathrm{k})$, where s_i denotes the number of variables in the i-cluster. To learn the clustering network, we relax the indicator matrix by squashing $\mathbf{C}$ into $(0,1)$. Specifically, we obtain the indicator matrix of $\mathbf{C}$ from $\mathbf{F}_\ell$, i.e., $\mathbf{C} = \sigma_c(\mathbf{F}_\ell \mathbf{W}_c + \mathbf{b}_c)$. We use the temperature softmax [78] as

the activation functions, i.e., $\sigma_c(z_i) = \frac{\exp(z_i/T_a)}{\sum_j \exp(z_j/T_a)}$ and reduce the temperature gradually. When T_a becomes smaller, the matrix $\mathbf{C}$ likely approximates $\{0,1\}$. In the experiments, we adopt a Gaussian decay function of $T_a = T_0 \max(\frac{1}{T_0}, exp(-\frac{e^2 \log(T_0)}{64}))$, where T_0 is the initial temperature and e denotes the number of epochs. The decay function is in a sigmoid-like shape which guarantees a low decay rate to avoid sharp fluctuations at the beginning and the end of the training. This treatment alleviates the training instability caused by the fluctuation of the learned cluster structures, accelerating training convergence. We further introduce an element-wise threshold function to filtrate low values in $\mathbf{C}$, formulated as $\text{Filter}(\mathbf{C}) = \frac{1+\text{sgn}(\mathbf{C}-\theta)}{2} C$, where $\theta = 1/(\sqrt{2}k)$, and sgn denotes the sign function. The threshold function results in a sparse relaxed matrix $\mathbf{C}$ that helps approximate the indicator matrix $\{0,1\}^{N\times k}$.

Let us denote the spectral clustering network as Ω, which includes DFT, the convolutional layer, and the fully connected layers. Theoretically, the Lipschitz continuity is proved on DFT in Corollary 6.1 and is also guaranteed on the convolutional and fully connected layers using spectral normalization [151]. Thus, we can easily prove the clustering network satisfying K-Lipschitz continuity, i.e., $\|\Omega(x_1) - \Omega(x_2)\| \leq K\|x_1 - x_2\|$ with $K > 0$. Here, Lipschitz continuity theoretically restricts the variation of variable scales, preserving the distances/similarities between time series variables in the clustering network (from $\mathbf{X}$ to $\mathbf{C}$). It guarantees that relevant variables with similar scales are clustered while irrelevant variables with different scales are apart in the clustering results. This facilitates the improvement of model generalization and network interpretability and alleviates the difficulty of modeling variable heterogeneity in a divide-and-conquer manner.

Corollary 6.1. *Given two time series* $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^N$*, then exist* $K > 0$ *satisfying* $\|\text{DFT}(\mathbf{x}_1) - \text{DFT}(\mathbf{x}_2)\| \leq K\|\mathbf{x}_1 - \mathbf{x}_2\|$.

Proof. Let us define DFT on univariate time series $\mathbf{x} = \{x_n\}_{n=0}^{N-1}$ below:

$$\mathbf{X} = \text{DFT}(\mathbf{x}) = \{X_k\}_{k=0}^{N-1} \tag{6.4}$$

where $X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn}$.
Now, given $\mathbf{X}_1 = \text{DFT}(\mathbf{x_1})$ and $\mathbf{X}_2 = \text{DFT}(\mathbf{x_2})$, let $\mathbf{y} = \mathbf{x}_1 - \mathbf{x}_2$ and $\mathbf{Y} = \mathbf{X}_1 - \mathbf{X}_2$, we have

$$\|\mathbf{y}\| = \|\mathbf{x}_1 - \mathbf{x}_2\| = \sqrt{\sum_{k=0}^{N-1} y_k^2} \tag{6.5}$$

$$\|\mathbf{Y}\| = \|\mathbf{X}_1 - \mathbf{X}_2\| = \sqrt{\sum_{k=0}^{N-1} |Y_k|^2} \tag{6.6}$$

According to Parseval's theorem [164], we can obtain $\|\mathbf{y}\| = \frac{1}{N}\|\mathbf{Y}\|$, thus we have

$$\|\mathbf{x}_1 - \mathbf{x}_2\| = \frac{1}{N}\|\mathbf{X}_1 - \mathbf{X}_2\| \geq \frac{1}{K}\|\operatorname{DFT}(\mathbf{x}_1) - \operatorname{DFT}(\mathbf{x}_2)\|$$

where $K \geq N$. Proved. ■

6.3.2 Clusterwise Forecasting Network

Building on the excellent performance of Transformer in modeling long-term dependencies for sequential data [104, 289], we propose a clusterwise forecasting network (shown on the right in Figure 6.2). It adopts the encoder-decoder architecture and Transformer to model intra-TS temporal correlations and inter-TS coupling relations. The network includes a **hete**rogeneous **emb**edding (HeteEmb) module, k-channel Transformers with **c**luster-aware **m**ulti-head **a**ttention (CMA), and clusterwise prediction layers.

6.3.2.1 Encoder-Decoder Architecture

We adopt the encoder-decoder architecture to represent the clusterwise time series inputs and generate clusterwise outputs for forecasting. Specifically, according to each cluster indicator vector $\mathbf{c}_i \in \mathbb{R}^{N\times 1}$ in $\mathbf{C}$, we distribute the L-step normalized time series $\mathbf{X}_L$ into k clusters and generate the clusterwise inputs, i.e., $\{\mathbf{X}_L^i = \mathbf{c}_i \odot \mathbf{X}_L \in \mathbb{R}^{n_i\times L}\}_{i=1}^k$. Here, $\odot$ denotes performing elementwise product and then selecting non-zero columns, and n_i denotes the number of time series variables in the i-th cluster where $\sum_{i=1}^k n_i = N$. The clusterwise inputs are fed into the HeteEmb module to generate the embeddings for each clusterwise input, followed by the encoder of the k-channel Transformers to generate a clusterwise feature map. Paired with the L-step inputs $\mathbf{X}_L$, we generate a concatenated input as follows:

$$\mathbf{X}_{de} = \operatorname{Concat}(\mathbf{X}_{token}, \mathbf{X}_0) \in \mathbb{R}^{N\times(L_{token}+D)} \tag{6.7}$$

where $\mathbf{X}_0 \in \mathbb{R}^{N\times D}$ is a placeholder for the target D-step values $\mathbf{Y}$ and only contains the target timestamp and position information. $\mathbf{X}_{token} \in \mathbb{R}^{N\times L_{token}}$ denotes the start token, where $L_{token} \leq L$ and $\mathbf{X}_{token} \subseteq \mathbf{X}_L$. It is a common and efficient technique in the Transformer-based models [48]. In parallel to the encoder, we obtain the clusterwise inputs of $\mathbf{X}_{de}$, embed the inputs by the HeteEmb module, and feed the embeddings to the

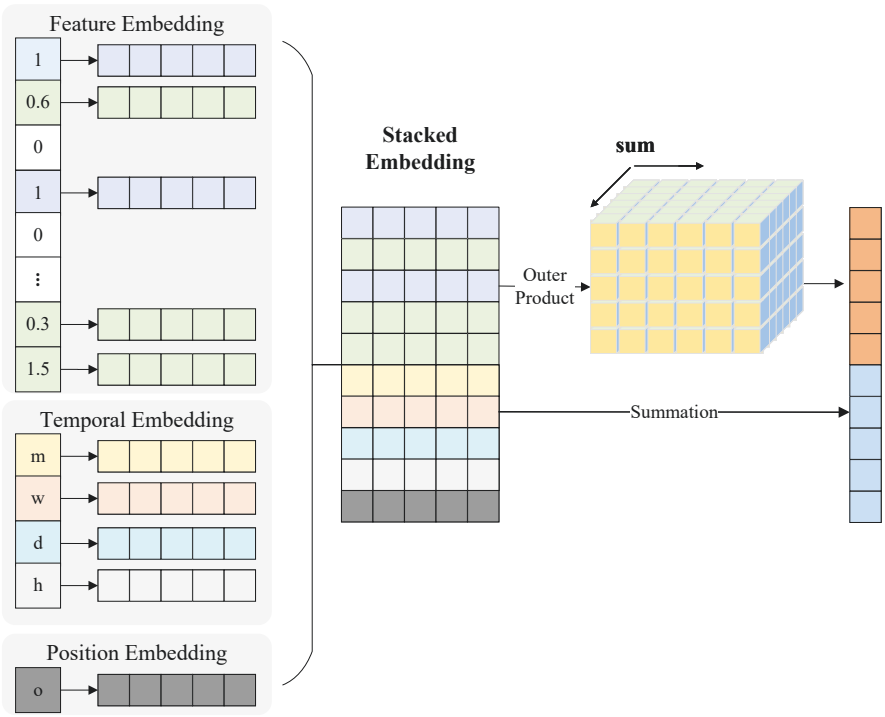


Figure 6.3: The heterogeneous embedding module.

We illustrate the temporal embedding with global timestamps of month (m), week (k), day (d), and hour (h) and the local position embedding, i.e., position (o) in the sequence.

masked CMA in the encoder of the k-channel Transformers. Following the masked CMA, another CMA module is introduced with the clusterwise feature map acting as its keys and values, and the outputs from the masked CMA as its queries. Note that the masked CMA is applied to prevent leftward information flow to avoid auto-regression [210]. Finally, we feed the decoder output to the clusterwise fully connected prediction layer and predict the target N-variable D-step values $\mathbf{Y}$ by one forward procedure instead of step by step.

6.3.2.2 Heterogeneous Embedding

We incorporate the hierarchical timestamps (week, month, and year) and position information with clusterwise MTS inputs. Considering the heterogeneity of the mixed inputs, e.g., the numeric time series, and categorical timestamps, it is unsuitable to concatenate the multivariate variables as a one-strand vector for calculating input representations. Inspired by [275] which considers the variable correlations, we propose a heterogeneous embedding module (HeteEmb) as shown in Figure 6.3 to perform variable embedding and

calculate the variable relations (within each cluster). Specifically, a variable embedding matrix $\mathbf{W}^e = \{\mathbf{w}_i^e\}_{i=1}^{N'} \in \mathbb{R}^{N' \times d_m}$ is introduced to embed the mixed input, where $N' > N$ is the number of variables after transformation since it includes the temporal and position variables and the categorical variables are binarized. Given the mixed input $\mathbf{x}_t^i \in \mathbf{X}_L^i$ and $\mathbf{x}_t^i \in \mathbb{R}^{n_i}$ at time step t in the i-th cluster, we have $x_{tj}^i \in \mathbf{x}_t^i$ denoting the j-th value in the n_i variables, then:

$$
\begin{aligned}
\mathbf{e}_{tj}^i &= x_{tj}^i \mathbf{w}_j^e \\
v_{tj}^i &= \sum_{p=1}^{n_i} \sum_{q=p+1}^{n_i} \left[\mathbf{E}_{t,j\cdot}^i \otimes \mathbf{E}_{t,j\cdot}^i \right] \Big|_{pq}
\end{aligned}
\tag{6.8}
$$

where $\mathbf{E}_t^i = \{\mathbf{e}_{tj}^i\}_{j=1}^{n_i} \in \mathbb{R}^{n_i \times d_m}$ denotes the stacked embeddings for $\mathbf{x}_t^i$ and $\mathbf{E}_{t,j\cdot}^i \in \mathbb{R}^{d_m}$ denotes its j-th row, and $\otimes$ denotes the outer product. Here, v_{tj}^i calculates the variable relations and results in an embedding vector $\mathbf{v}_t^i = \{v_{tj}^i\}_{j=1}^{d_m} \in \mathbb{R}^{d_m}$ for the i-th cluster at time step t. Note that the variable coupling relation calculation is time-efficient with the time complexity of $\mathcal{O}(Nd_m)$, referring to Factorization Machine [171]. The calculation is pairwise and proved more efficient than feed-forward neural networks to capture the second-order correlations between time series variables, timestamps, and position information [12].

In addition, we sum over $\mathbf{E}_t^i$ to generate a first-order correlation, i.e., $\mathbf{u}_t^i = \sum_{i=1}^{n_i} \mathbf{E}_t^i$, and concatenate the second-order and first-order correlations to generate the final embedding $\mathbf{z}_t^i = \text{Concat}(\mathbf{v}_t^i, \mathbf{u}_t^i)$ for $\mathbf{x}_t^i$. Accordingly, we obtain the clusterwise embeddings $\mathcal{Z} = \{\mathbf{Z}_i\}_{i=1}^k$ for k clusters, with $\mathbf{Z}_i = \text{Stack}\left(\{\mathbf{z}_t^i\}_{t=1}^L\right)$ of L-step time series for the encoder. We obtain the clusterwise embeddings for the inputs of the decoder in a similar manner. Intuitively, the first-order calculation can be considered as one fully connected layer that has weights $\mathbf{W}^e$ and acts on the mixed input $\mathbf{x}_t^i$. The HeteEmb module projects the clusterwise inputs with different numbers of variables into d_m-sized embeddings respectively, subtly shielding the difference in the number of variables in different clusters, i.e., $\{n_i\}_{i=1}^k$. The combination of first-order and second-order correlations facilitates capturing the variable heterogeneity and sufficient information.

6.3.2.3 Cluster-aware Multihead Attention

The self-attention used in Transformer is the scaled dot-product attention defined on the tuple input (queries $\mathbf{Q}$, keys $\mathbf{K}$, values $\mathbf{V}$) [210], where queries and keys have d_k dimensions and values have d_v dimensions. The dot-product of queries with keys is scaled by $\frac{1}{\sqrt{d_k}}$ and normalized to obtain the weights using softmax functions. Then, the

outputs of self-attention are calculated by attending to values referring to the weights, formulated as follows:

$$\mathscr{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{6.9}$$

Under the multihead settings, d_m-dimensional queries, keys and values are projected to d_k, d_k, and d_v dimensions for h times by different linear projections respectively. Multihead attentions are then performed on these tuple inputs in parallel and yield $d_v * h$ dimensional output values, generally holding $d_v = d_k = d_m/h$.

In addition to the standard Transformer, we can apply other efficient and low-storage self-attention mechanisms, such as LogTrans [118], Reformer [104], Informer [289], for fast computation, especially on long time series. Specifically, we adopt efficient locality-sensitive hashing (same as Reformer) and ProbSparse attention (same as Informer) to replace dot-product attention in Cospectrumer in the experiments. For simplicity, we introduce our cluster-aware multihead attention based on vanilla self-attention (i.e., Equation 6.9) in the following which can be easily extended to efficient attention mechanisms via updating the queries (refer to Equation 6.10).

In the k-channel Transformers, we introduce cluster-aware attention to learn cluster-wise correlations to promote forecasting performance. For the i-th cluster, we treat the information from other clusters as the context of the i-th cluster, denoted as $\mathscr{Z}_{\bar{i}} = \mathscr{Z} - \{\mathbf{Z}_i\}$. The context $\mathscr{Z}_{\bar{i}}$ is flattened to the size of $L \times (k-1)d_m$ along the cluster dimension and aligned to $\mathbf{Z}_i$ according to time steps. It is utilized to enhance the queries in the attention calculation. Given the clusterwise embeddings of L-step time series, $\mathscr{Z} = \{\mathbf{Z}_1, \mathbf{Z}_2, \cdots, \mathbf{Z}_k | \mathbf{Z}_i \in \mathbb{R}^{L \times d_m}\}$, we have

$$\begin{aligned}
\text{CMA}(\mathscr{Z}) &= \text{Stack}(\text{channel}_1, \cdots, \text{channel}_k) \times_T \mathbf{W}^O \\
\text{channel}_i &= \text{Concat}(\text{head}_i^1, \cdots, \text{head}_i^h) \\
\text{head}_i^j &= \mathscr{A}(\mathbf{Z}_i\mathbf{W}_j^Q + \mathscr{Z}_{\bar{i}}\mathbf{W}_j^C, \mathbf{Z}_i\mathbf{W}_j^K, \mathbf{Z}_i\mathbf{W}_j^V)
\end{aligned} \tag{6.10}$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^C \in \mathbb{R}^{(k-1)d_m \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_v}$ and $\mathbf{W}^O \in \mathbb{R}^{k \times d_v \times d_m}$ are projection parameters, and head_i^j denotes the j-th single head attention in the i-th cluster. The operator $\times_T$ denotes the tensor multiplication. We feed each clusterwise input into one channel for separate attention calculation and supplement the context information upon queries to achieve context-aware (cluster-aware) attention weights. Subsequently, we apply the operator of Stack to stack the attention outputs of each channel and parallel feed cluster-aware attention outputs to feed-forward networks followed by

layer normalization [5] and residual connections channel-by-channel, guaranteeing the information isolation between the channels. The calculation is formulated as:

$$\mathcal{Z}^l = \text{LayerNorm}(\mathcal{Z}^{l-1} + \text{CMA}(\mathcal{Z}^{l-1})) \tag{6.11}$$

where $\mathcal{Z}^l$ denotes the inputs of the l-th attention layer, and $\mathcal{Z}^0 = \mathcal{Z}$. Then $\mathcal{Z}^l$ is fed into the feed-forward networks with two linear transformations by a ReLU activation in between:

$$\mathcal{Z}^l = \text{FFN}(\mathcal{Z}^l) = \text{ReLU}(\mathcal{Z}^l \times_T \mathbf{W}_1^l + \mathbf{b}_1^l) \times_T \mathbf{W}_2^l + \mathbf{b}_2^l \tag{6.12}$$

where $\mathbf{W}_1 \in \mathbb{R}^{k \times d_m \times d_{ff}}$ and $\mathbf{W}_2 \in \mathbb{R}^{k \times d_{ff} \times d_m}$. Thus, we obtain the clusterwise feature map $\mathcal{Z}^{\ell_e} \in \mathbb{R}^{L \times k \times d_m}$ from an ℓ_e-layer encoder and the clusterwise output $\mathcal{Z}^{\ell_d} = \{\mathbf{Z}_i^{\ell_d} \in \mathbb{R}^{(L_{token}+D) \times d_m}\}_{i=1}^k$ from an ℓ_d-layer decoder.

6.3.2.4 Prediction Layers and Loss Function

The outputs of the decoder are projected to the predictions, denoted by $\hat{\mathbf{Y}}$, by clusterwise fully connected prediction layers in a clusterwise manner. In other words, the outputs of the i-th channel of the decoder are projected to predict the variables located in the i-th cluster. Given $\mathcal{Z}^{\ell_d}$, we have:

$$\hat{\mathbf{Y}} = \text{Concat}(\{\hat{\mathbf{Y}}_i\}_{i=1}^k),\ s.t.,\ \hat{\mathbf{Y}}_i = \mathbf{Z}_i^{\ell_d}(\mathbf{c}_i \odot \mathbf{W_o})^T + \mathbf{b}_i$$

where $\mathbf{W}_o \in \mathbf{R}^{N \times d_m}$ and $\mathbf{b}_i \in \mathbf{R}^{1 \times n_i}$ denote the weights and bias of the prediction layer, and $\hat{\mathbf{Y}}_i \in \mathbf{R}^{(L_{token}+D) \times n_i}$.

We adopt the mean square error loss function (denoted by MSE) to calculate the difference between prediction $\hat{\mathbf{Y}}$ and the target values $\mathbf{Y}$, i.e., $\mathscr{L}_{MSE} = \text{MSE}(\mathbf{Y}, \hat{\mathbf{Y}})$. Combining the K-means objective in Equation (6.3), the final loss function is given as:

$$\mathscr{L}(\Omega, \Theta) = \mathscr{L}_{MSE}(\Phi) + \lambda \mathscr{L}_{KM}(\Theta) \tag{6.13}$$

where Ω and Θ denote the parameters of the clustering network and the forecasting network respectively. λ denotes the weight for the K-means loss. Since the update of Ω may dramatically influence the learning of Θ, we perform layer normalization on $\mathbf{F}_\ell \mathbf{F}_\ell^T$ and fix the value of $\mathbf{F}_\ell \mathbf{F}_\ell^T$ in calculating gradients in Equation (6.3).

6.4 Experiments and Evaluation

We conduct extensive experiments to investigate the performance of Cospectrumer in terms of the five questions:

Q1. How accurate is Cospectrumer compared with the SOTA deep neural MTS baselines?

Q2. What is the model complexity of Cospectrumer?

Q3. To what extent does the design of Cospectrumer, e.g., the clustering network, the CMA module, and the HeteEmb module, improve the performance?

Q4. How does the hyperparameter, e.g., the number of clusters, affect the performance of Cospectrumer?

Q5. What is the impact of TS clustering on forecasting?

6.4.1 Datasets

We evaluate the proposed Cospectrumer on four real-world benchmark datasets covering different mainstream MTS forecasting applications: electricity, weather, energy, and traffic. (1) *Electricity*[1]: this dataset is a collection of electricity consumption data of 370 clients from 2011 to 2015. Considering the missing data, we select the consumption data of 315 clients from 2012 to 2015 and convert the data into hourly consumption. (2) *Weather*[2]: we select three-year data from 2018 to 2020 from WS Beutenberg which was recorded every 10 minutes and contains 21 meteorological indicators, such as air temperature, and humidity. (3) *Solar*[3]: this dataset is a collection of solar power production records in 2006, which was sampled every 10 minutes from 137 PV plants in Alabama, USA. (4) *traffic*[4]: this dataset is a collection of hourly data from the California Department of Transportation and contains the occupancy rate of 862 lanes on a San Francisco highway from January in 2015. The dataset details are summarised in Table 6.1. Following a common protocol [289], we split the datasets into training, validation, and test sets in a chronological order using the ratio of 7:1:2. In addition, we adopt the raw input for the clustering network and perform zero-mean normalization (standard normalization) on the input of the forecasting network for all datasets.

[1]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[2]https://www.bgc-jena.mpg.de/wetter/
[3]https://www.nrel.gov/grid/solar-power-data.html
[4]http://pems.dot.ca.gov

Table 6.1: Statistics of four multivariate time series data.

Datasets	#Samples	#Variates	Granularity	Start time
Electricity	26304	315	1hour	1/1/2012
Weather	157773	21	10min	1/1/2018
Solar	52560	137	10min	1/1/2006
Traffic	17533	862	1hour	1/1/2015

6.4.2 Experimental Details

Below, we introduce the baselines and detail their experimental settings against our model.

Baselines. We compare Cospectrumer with eight SOTA MTS forecasting models and Transformer-based models:

- *LSTNet* [111]: A SOTA RNN-based MTS forecasting framework capturing the hidden long- and short-term dependencies of time series using LSTM.
- *MLCNN* [37]: A SOTA CNN-based MTS forecasting model applying CNN to extract multilevel features and employing LSTMs to capture hidden temporal correlations for multiple predictive tasks.
- *StemGNN* [18]: A spectral- and GCN-based method introducing graph Fourier transform and DFT to jointly capture inter-series correlations and temporal dependencies.
- *Reformer* [104]: A memory-efficient and fast Transformer variant for long sequence prediction replacing dot-product attention by the one using locality-sensitive hashing and using reversible residual layers instead of standard residuals.
- *LogTrans* [118]: A Transformer variant introducing causal convolution to perceive local context into an attention mechanism and involving a LogSparse self-attention to reduce memory usage.
- *Informer* [289]: The latest Transformer-based model introducing a ProbSparse mechanism and a distilling operation for time- and memory-efficient self-attention calculation to predict long sequence time series at one forward operation.
- *Autoformer* [243]: A SOTA Transformer-based model proposing a decomposition architecture with an efficient and accurate auto-correlation mechanism for long-term MTS forecasting.

- *FEDformer* [290]: A SOTA Transformer-based model based on Autoformer introducing an attention mechanism with low-rank approximation in the frequency domain and a mixture of experts decomposition to control the distribution shifting.

These SOTA MTS forecasting baselines are chosen carefully for purposeful comparisons: LSTM-, CNN-, and GNN-based methods are compared to investigate the superiority of Transformer-based methods; Transformer- and spectral-based (StemGNN) methods are selected to investigate the effectiveness of Cospectrumer w.r.t. its clustering and forecasting networks. Traditional MTS methods are not considered in the experiments since recent deep learning studies demonstrate their excellent prospects and performance for MTS and Cospectrumer belongs to this neural family. In addition, since this is the first attempt to adaptively learn time series clustering to improve MTS forecasting, there is no similar clustering-based method for comparison.

Cospectrumer variants. For the ablation study, we explore the performance of the spectral clustering network and Transformers by creating four variants of Cospectrumer: 1) an efficient variant Cospectrumer$^{\mathscr{P}}$ employing the ProbSparse self-attention mechanism; 2) a fixed cluster scheme variant Cospectrumer†, i.e., the Cospectrumer with the fixed cluster indicator matrix $\mathbf{C}$ obtained from K-means clustering in advance; 3) a decoupled variant Cospectrumer‡ without considering the cluster-aware attention mechanism, i.e., without $\mathscr{Z}_{\tilde{i}}\mathbf{W}_j^C$ in Equation (6.10); 4) a plain embedding Cospectrumer$^{\sharp}$ without the heterogeneous embedding module; 5) a variant Cospectrumer$^{\mathscr{S}}$ with normalized MTS inputs in the clustering network.

Experiment Settings. 1) *parameter settings:* Our proposed model is trained using the ADAM optimizer [212] with an initial learning rating of 10^{-4} and a batch size of 64. Each training process is stopped early with a total of 10 epochs. We tune the hyperparameters on the validation set by grid searching the number of clusters k over $\{2,4,6,8,10\}$, the length of embeddings d_m over $\{32,64,128,256\}$ and the weight $\lambda \in \{0.1,0.3,0.5,0.7,0.9\}$. We set $d_m = 256$ and $\lambda = 0.1$ for all datasets where Cospectrumer achieves the best performance on the validation set. In addition, we adopt the efficient locality-sensitive hashing for long-term forecasting (e.g., $L \geq 96$ or $D \geq 48$) and set: i) the downsampling size $L_d = 1000$ and the number of fully connected layers $\ell = 1$ in the spectral clustering network; ii) the initial temperature $T_0 = 16$; iii) $\ell_e = 2$ and $\ell_d = 1$; iv) the number of attention heads $h = 8$ and the dimension $d_{ff} = 1024$ in the feedforward layers. 2) *baselines settings:* To guarantee a fair comparison, we set the comparison methods with author-recommended settings and use the same network parameters for the Transformer-based methods. The best parameters for all comparative

models are chosen by carefully tuning the author-recommended parameter ranges on the validation set. In addition, we choose the length of inputs $L = 96$, the length of the start token $L_{token} = 48$, and the length of prediction $D = 24$ for all comparative methods if not specified. 3) *evaluation:* We use two evaluation metrics, i.e., RMSE and MAE. We repeat all experiments twice and report the average evaluation results. 4) *platform:* All experiments were implemented in PyTorch and were conducted on two NVIDIA 1080 12G GPUs.

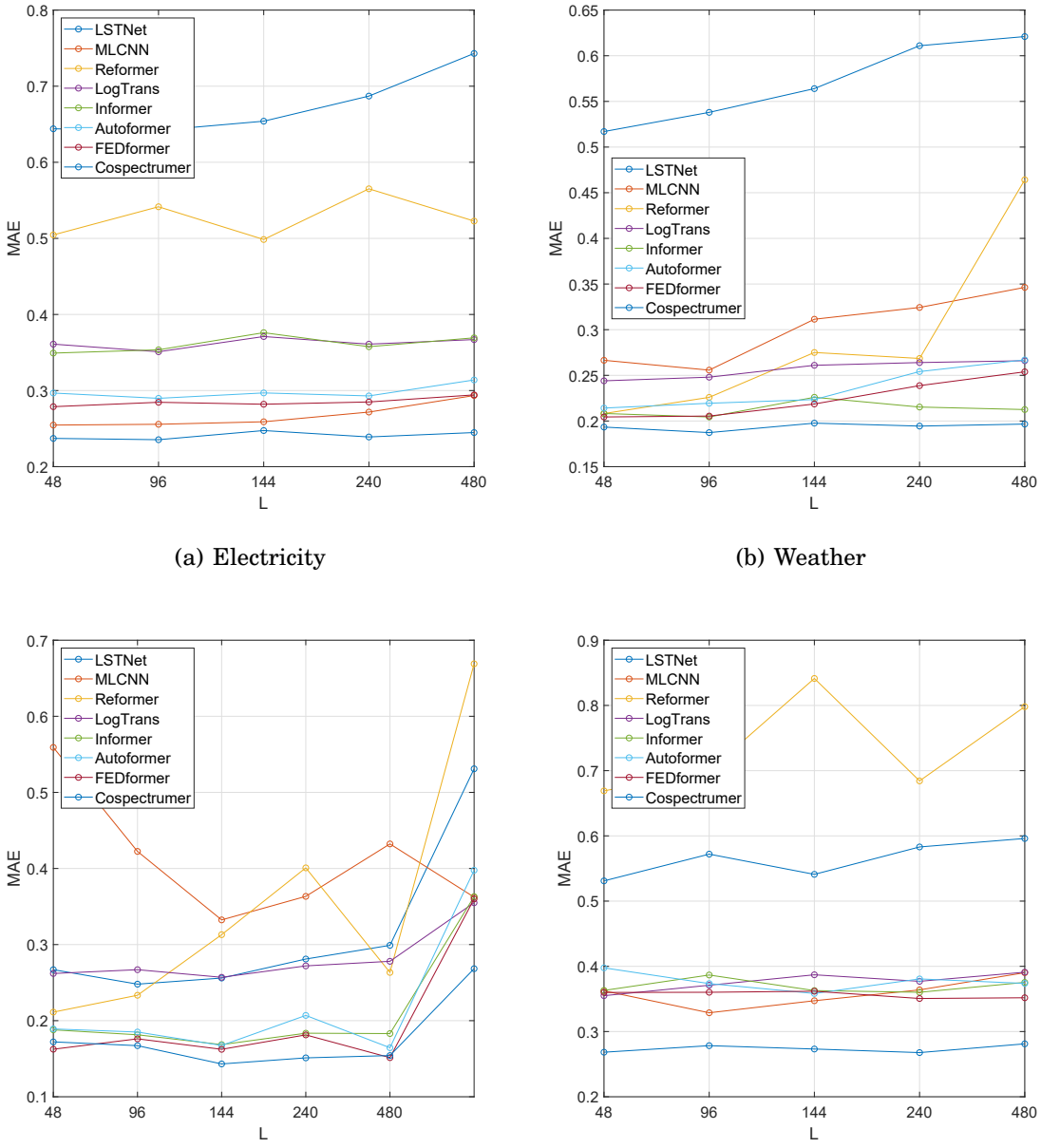
Table 6.2: Quantitative results in terms of MAE and RMSE with prediction lengths $D \in \{24, 48, 96, 240, 720\}$, $L = 96$, and $L_{token} = 48$.

Datasets		LSTNet		MLCNN		StemGNN		Reformer		LogTrans		Informer		Autoformer		FEDformer		Cospectrumer		Imp. (%)	
Metrics		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE		
Electricity	24	0.642	0.783	0.750	1.006	**0.233**	0.344	0.542	0.731	0.351	0.485	0.354	0.493	0.290	0.411	0.285	0.404	0.235	**0.336**	- -	2.38*
	48	0.660	0.811	0.981	1.271	0.256	**0.378**	0.612	0.812	0.357	0.507	0.368	0.519	0.305	0.434	0.297	0.428	**0.250**	0.399	2.40*	- -
	96	0.674	0.836	1.027	1.209	0.278	0.409	0.693	0.897	0.361	0.511	0.371	0.521	0.310	0.442	0.308	0.443	**0.262**	**0.406**	6.11*	0.74*
	240	0.687	0.864	1.045	1.322	NA	NA	0.829	1.018	0.368	0.517	0.386	0.541	0.403	0.564	0.332	0.468	**0.271**	**0.441**	22.5*	6.12*
	720	0.823	1.026	1.124	1.521	NA	NA	0.853	1.061	0.402	0.637	0.392	0.584	0.612	0.830	0.360	0.508	**0.298**	**0.462**	20.8*	9.96*
Weather	24	0.538	0.734	0.262	0.737	0.238	0.598	0.226	0.573	0.248	0.583	0.204	0.569	0.219	0.573	0.193	0.546	**0.187**	**0.531**	3.21*	2.82*
	48	0.552	0.743	0.385	0.876	0.275	0.641	0.471	0.809	0.312	0.610	0.256	0.623	0.257	0.619	0.254	0.617	**0.245**	**0.596**	3.67*	2.35*
	96	0.574	0.766	0.660	1.216	0.333	0.700	0.476	0.817	0.412	0.665	0.309	0.692	0.313	0.704	0.305	0.686	**0.281**	**0.639**	8.54*	4.07*
	240	0.561	0.751	1.125	1.672	NA	NA	0.591	0.952	0.526	0.771	0.390	0.766	0.386	0.763	0.384	0.760	**0.357**	**0.734**	7.56*	3.54*
	720	0.621	0.837	1.655	2.561	NA	NA	0.607	0.963	0.672	0.917	0.461	0.832	0.459	0.823	0.454	0.820	**0.421**	**0.782**	7.84*	4.86*
Solar	24	0.248	0.489	0.309	0.808	0.168	**0.309**	0.233	0.362	0.267	0.510	0.181	0.332	0.185	0.338	0.176	0.311	**0.167**	0.311	0.60	- -
	48	.0.314	0.607	0.784	1.485	**0.221**	0.419	0.341	0.535	0.311	0.579	0.257	0.446	0.243	0.433	0.238	0.426	0.232	**0.407**	- -	2.95*
	96	0.401	0.673	1.168	2.013	0.274	0.470	0.413	0.632	0.385	0.641	0.251	0.450	0.254	0.496	0.247	0.457	**0.227**	**0.424**	8.81*	6.13*
	240	0.442	0.711	1.129	2.168	NA	NA	0.447	0.683	0.431	0.703	0.263	0.472	0.264	0.474	0.261	0.470	**0.242**	**0.457**	7.85*	2.84*
	720	0.623	0.875	1.442	2.426	NA	NA	0.534	0.761	0.562	0.810	0.273	0.501	0.273	0.501	0.274	0.497	**0.253**	**0.477**	7.91*	4.19*
Traffic	24	0.572	0.756	0.874	1.353	0.292	0.709	0.700	1.076	0.371	0.783	0.387	0.824	0.374	0.758	0.360	0.743	**0.278**	**0.703**	5.04*	0.85*
	48	0.602	0.821	0.966	1.428	0.297	**0.717**	0.832	1.195	0.377	0.802	0.409	0.855	0.389	0.779	0.365	0.754	**0.293**	0.744	1.37*	- -
	96	0.641	0.897	1.010	1.459	0.317	0.741	0.829	1.231	0.384	0.827	0.379	0.824	0.416	0.820	0.369	0.770	**0.284**	**0.721**	11.6*	2.77*
	240	0.692	0.976	1.066	1.504	NA	NA	0.825	1.232	0.418	0.872	0.411	0.864	0.437	0.836	0.373	0.804	**0.313**	**0.778**	21.1*	3.34*
	720	0.811	1.243	1.103	1.527	NA	NA	0.821	1.235	0.442	0.906	0.476	0.948	0.478	0.880	0.386	0.813	**0.325**	**0.787**	18.8*	3.30*

6.4.3 Performance Evaluation

We evaluate Cospectrumer against the baselines under different prediction lengths on four datasets in Table 6.2. The best results are highlighted in bold, and the best baseline results are underlined. NA denotes that results are not available due to running out of memory. *Imp.* shows the performance improvement of Cospectrumer over the best baseline in terms of MAE and RMSE respectively. Herein, - - denotes negative improvement and $*$ indicates statistically significant improvement using the two-sided t-test with $p < 0.05$. As shown in Table 6.2, Cospectrumer achieves the SOTA performance compared with the baselines especially on long-term forecasting. Specifically, Cospectrumer outperforms all the baselines and achieves $6.11\% - 22.5\%$ MAE improvement and $0.74\% - 9.96\%$ RMSE improvement over the baselines on all datasets for the prediction length $D \in \{96, 240, 720\}$. When $D \in \{24, 48\}$, Cospectrumer also achieves competitive forecasting performance and surpasses all the baselines except for StemGNN. Especially on the Weather dataset, Cospectrumer outperforms all the baselines for all prediction lengths and makes a 3.21% MAE improvement and a 2.35% RMSE improvement at least. Table 6.2 shows that Cospectrumer is much better than the RNN- and CNN-based methods, demonstrating the benefit of Transformers in capturing the long-term dependencies in sequential data. Compared to RNN and CNN, Transformer-based networks do not impose a strong sequential or spatial hypothesis on MTS and perform much better and more flexibly in capturing inter-/intra-TS dependencies. The GNN-based StemGNN achieves desirable performance on Electricity, Solar and Traffic although it is unsuitable for long-term prediction. This is because StemGNN introduces graph Fourier transform and DFT to model inter-TS correlations and temporal dependencies respectively and facilitates the extraction of periodic spectral representation. In addition, Cospectrumer achieves better performance over the Transformer-based methods (including the SOTA Informer, Autoformer and FEDformer) on all datasets and at all prediction lengths. The results verify the benefits of time series clustering for MTS forecasting and indicate the efficacy of the spectral clustering network and clusterwise forecasting network in Cospectrumer for capturing intra-TS temporal patterns and inter-TS couplings.

We further investigate the forecasting performance of the comparative methods at different input lengths, with the results shown in Figure 6.4. Cospectrumer performs stably over different input lengths, especially for long inputs, which can be attributed to adopting Transformer as the backbone to model intra-TS temporal patterns. In addition, Cospectrumer achieves better accuracy and more stable performance over the baselines, which confirms the results in Table 6.2 that the architecture of Cospectrumer helps



(a) Electricity

(b) Weather

(c) Solar

(d) Traffic

Figure 6.4: Performance comparison of different input lengths $L \in \{48, 96, 144, 240, 480\}$ with $D = 24$ and $L_{token} = 48$.

improve the forecasting performance on MTS.

Table 6.3: Statistics of model complexity in terms of training/test time costs and model parameter volume.

Method	Parameter Volume (M)	Time (Second/Epoch)	Time (Second)
LSTNet	0.22	2.75	0.55
MLCNN	0.25	13.12	0.73
StemGNN	69.88	559.45	56.32
Reformer	17.53	49.59	4.65
LogTrans	17.29	96.34	7.31
Informer	14.4	18.91	3.33
Autoformer	14.94	30.92	4.69
FEDFormer	18.82	61.41	6.16
Cospectrumer	19.73	115.1	8.57
Cospectrumer$^{\mathscr{P}}$	17.36	63.88	6.22

6.4.4 Model Complexity

To investigate the model complexity of Cospectrumer, we verify the impact of its parameter volume and time cost. Specifically, we compare Cospectrumer ($k = 8$) with other Transformer-based methods on Traffic (with the highest dimension) under the same parameter settings as follows: $L = 96$, $D = 24$, $L_{token} = 48$, $d_m = 512$, $d_{ff} = 2048$, $\ell = 2$ and $\ell_d = 1$. In addition, we adopt the recommended settings for LSTNet, MLCNN, and StemGNN.

The statistics of the time costs and parameter volumes are shown in Table 6.3. We investigate 1) all the comparative baselines; 2) Cospectrumer using locality-sensitive hashing to replace dot-product attention (same as Reformer); 3) the efficient variant Cospectrumer$^{\mathscr{P}}$ using the ProbSparse self-attention mechanism (same as Informer). From the experimental results, we observe 1) the parameter volumes of Cospectrumer are larger than the Transformer-based methods, while Cospectrumer$^{\mathscr{P}}$ using the ProbSparse mechanism achieves a comparable parameter volume relative to Reformer, LogTrans, and FEDformer; 2) the time costs of Cospectrumer are larger than the Transformer-based methods, while Cospectrumer$^{\mathscr{P}}$ with the ProbSparse mechanism saves a lot of training/test time cost and is competitive compared to LogTrans and FEDformer in terms of time efficiency; 3) StemGNN needs a much larger parameter volume and higher time costs than the SOTA Transformer-based methods despite its competitive forecasting performance; 4) compared to LSTM- and CNN-based methods, the Transformer-based methods and StemGNN have a much higher model complexity but achieve much better forecasting performance.

Table 6.4: Ablation study of Cospectrumer in terms of its different input lengths $L \in \{48, 96, 144\}$.

Datasets	Metrics	Electricity			Weather		
		48	96	144	48	96	144
Cospectrumer	MAE	**0.237**	**0.235**	**0.247**	**0.193**	**0.187**	**0.198**
	RMSE	**0.341**	**0.336**	**0.366**	**0.542**	**0.531**	**0.554**
Cospectrumer^{P}	MAE	0.240	0.236	0.249	0.195	**0.187**	0.199
	RMSE	0.344	0.338	0.368	0.543	**0.530**	**0.554**
Cospectrumer†	MAE	0.254	0.249	0.258	0.197	0.188	0.203
	RMSE	0.367	0.358	0.379	0.549	0.533	0.561
Cospectrumer‡	MAE	0.247	0.241	0.252	0.208	0.195	0.214
	RMSE	0.359	0.344	0.373	0.563	0.541	0.569
Cospectrumer$^{\sharp}$	MAE	0.248	0.243	0.257	0.214	0.204	0.220
	RMSE	0.362	0.347	0.375	0.573	0.557	0.579
Cospectrumer$^{\mathscr{S}}$	MAE	0.246	0.238	0.252	0.203	0.194	0.205
	RMSE	0.353	0.342	0.374	0.552	0.540	0.563
Datasets	Metrics	Solar			Traffic		
		48	96	144	48	96	144
Cospectrumer	MAE	**0.172**	**0.167**	**0.143**	**0.268**	**0.278**	**0.273**
	RMSE	**0.323**	**0.311**	**0.295**	**0.702**	**0.703**	**0.711**
Cospectrumer^{P}	MAE	0.179	0.168	0.145	0.271	**0.278**	**0.272**
	RMSE	0.332	0.313	0.296	0.703	0.704	**0.709**
Cospectrumer†	MAE	0.182	0.173	0.152	0.285	0.297	0.291
	RMSE	0.338	0.321	0.306	0.742	0.765	0.757
Cospectrumer‡	MAE	0.181	0.173	0.155	0.277	0.286	0.282
	RMSE	0.340	0.322	0.309	0.711	0.734	0.739
Cospectrumer$^{\sharp}$	MAE	0.187	0.177	0.161	0.273	0.283	0.278
	RMSE	0.348	0.330	0.314	0.708	0.716	0.712
Cospectrumer$^{\mathscr{S}}$	MAE	0.180	0.172	0.151	0.276	0.284	0.281
	RMSE	0.334	0.321	0.303	0.716	0.719	0.722

Naturally, the clustering network and the multi-channel (cluster) computation bring extra parameters and time costs to Cospectrumer. Fortunately, Cospectrumer improves MTS forecasting and provides a new perspective of interpretability, while its parameter volume and time complexity are not greatly expanded. This is attributed to our following designs: (1) adopting an offline DFT to obtain frequency features; (2) using convolutional layers to downsample frequency features to avoid an overlong feature map; (3) sharing parameters between different clusters; (4) making a clusterwise prediction using the clusterwise fully connected prediction layers; and (5) employing the efficient self-attention mechanism for long sequence prediction.

6.4.5 Ablation Study

To investigate the effectiveness of the constitutional modules of Cospectrumer, we compare Cospectrumer with its three variants introduced in Section 6.4.2 under different input lengths. We choose the number of clusters $k = 4$ for Weather and $k = 8$ for Electricity, Solar, and Traffic, and set $D = 24$ and $L_{token} = 48$ for all datasets. The comparative results are shown in Table 6.4 where the best results are highlighted in bold, and the best results for different input lengths are underlined. Cospectrumer outperforms its variants on all datasets under different input lengths, demonstrating the applicability of Cospectrumer and the effectiveness of the components, i.e., the clustering network, CMA mechanism, and HeteEmb module, in achieving the SOTA results for Cospectrumer.

We adopt the ProbSparse self-attention mechanism in Cospectrumer$^{\mathscr{P}}$ and evaluate its performance for all input lengths $L \in \{48, 96, 144\}$. Table 6.2 shows the following: 1) Cospectrumer outperforms Cospectrumer$^{\mathscr{P}}$ on $L = 48$ for all datasets. This shows that ProbSparse self-attention gains efficiency with a slight performance loss; 2) Cospectrumer$^{\mathscr{P}}$ achieves desirable and even better performance relative to Cospectrumer, indicating ProbSparse self-attention is effective and competitive compared with the efficient attention with locality-sensitive hashing; 3) Cospectrumer$^{\mathscr{P}}$ shows superior performance compared with the baselines (refer to Table 6.2), confirming the effectiveness of the clustering network for promoting the forecasting performance. The results demonstrate the applicability of the architecture of Cospectrumer to state-of-the-art attention mechanisms.

In addition, we pre-compute the indicator matrix by K-means on the DFT output in Cospectrumer†. First, the superiority of Cospectrumer† over the baselines in Table 6.2 indicates the contribution of time series clustering to MTS forecasting. Then, we observe that adaptively learning to cluster (Cospectrumer) is better than pre-computed clustering (Cospectrumer†), especially on the high-dimensional time series in Electricity and Traffic. This is because: 1) it is difficult to perform clustering on high-dimensional data; 2) Cospectrumer learns proper clustering to promote MTS forecasting. The results indicate that the forecasting objective is potentially beneficial to improving time series clustering, inspiring us to introduce forecasting objectives into time series clustering tasks.

In Cospectrumer‡, we directly calculate the multi-head attention for each cluster separately without considering the cluster information as context. The improvement of Cospectrumer over Cospectrumer‡ shows that the CMA module effectively captures inter-cluster time series correlations to improve the prediction. Intuitively, the intra-

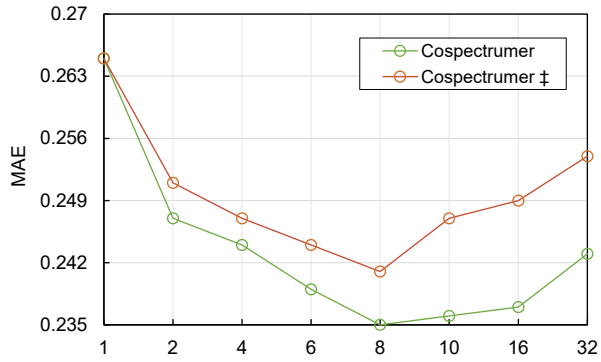
cluster time series correlation is much closer than the inter-cluster one, which, however, still plays a positive role in prediction. This is because Cospectrumer extracts clusterwise embeddings using HeteEmb to learn cluster-aware representation by CMA for future forecasting. In addition, Cospectrumer‡ performs slightly better than Cospectrumer†, indicating that time series clustering is able to divide and correlate target-irrelevant/relevant series, thus achieving better MTS forecasting than aimlessly modeling inter-TS correlations.

We adopt the uniform input representation method in Informer [289] to replace the HeteEmb module in Cospectrumer$^{\sharp}$. From Table 6.4, the results show that Cospectrumer$^{\sharp}$ performs worse than Cospectrumer, especially on Weather. This is attributed to the heterogeneous embedding module in capturing the heterogeneity of MTS by variable embeddings and learning second-order variable correlations. This treatment can learn a more informative representation for MTS and is more effective than the multilayer perceptron [275]. In addition, we investigate the performance of Cospectrumer with normalized inputs in the clustering network (i.e., Cospectrumer$^{\mathscr{S}}$). Cospectrumer achieves better performance than Cospectrumer$^{\mathscr{S}}$, indicating variable scales and distributions facilitate effective time series clustering and accordingly have a positive impact on capturing inter-/intra-TS couplings and addressing the heterogeneity issue.

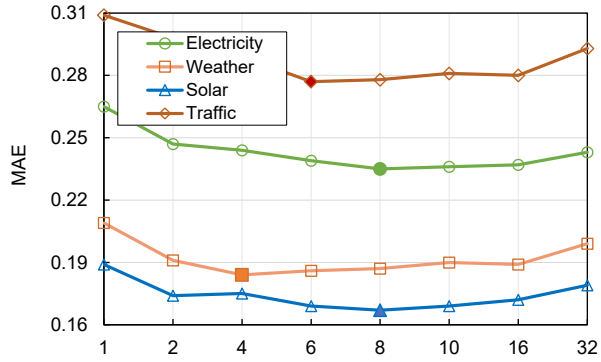
6.5 Parameter Sensitivity

We further perform experiments to investigate the parameter sensitivity of Cospectrumer. Four experiments investigate the impact of the number of clusters k, the embedding dimension d_m, the temperature T_0, and the learning weight λ, respectively. All the experiments are conducted under the settings: $k = 8$ (except for the experiment on k), $L = 96$, $D = 24$, and $L_{token} = 48$.

As shown in Figure 6.5(a), we first evaluate Cospectrumer and Cospectrumer‡ under $k \in \{1, 2, 4, 6, 8, 10, 16, 32\}$ on Electricity. The results show that: 1) Cospectrumer outperforms Cospectrumer‡ under different numbers of clusters; 2) Cospectrumer degenerates into Cospectrumer‡ at $k = 1$ such that the two models achieve equal performance. We further investigate the impact of the number of clusters on Cospectrumer. The MAEs of Cospectrumer on different datasets are reported in Figure 6.5(b) where the best result on each dataset is highlighted with a solid mark. From the results, we can observe that: 1) the best number of clusters (denoted k^*) varies across different datasets. Specifically, k^* is set to 4 on Weather, 6 on Traffic, and 8 on Solar and Electricity, respectively; 2)



(a) Cospectrumer and Cospectrumer‡ on Electricity



(b) Cospectrumer on four different datasets

Figure 6.5: Impact analysis of the number of clusters.

the performance of Cospectrumer is improved when the number of clusters k increases (from 1 to k^*), indicating the clustering network is effective in promoting the forecasting performance; 3) the performance will not degrade significantly when $k > k^*$ and k keeps increasing. This may be because the spectral clustering network can adaptively learn the number of clusters to guarantee effective time series clustering (refer to Section 6.6 for a more detailed explanation). The results indicate that a relatively larger number of clusters (e.g., $4 < k < 16$) may result in more desirable performance. As k increases further, it brings more training parameters and costs to the clustering network, subsequently leading to the inferior performance of Cospectrumer.

The experiment results of the six comparative methods under different embedding dimensions are shown in Figure 6.6. Cospectrumer and its variant Cospectrumer† perform stably and achieve consistently better MAE than the baselines. This confirms that the proposed clustering network and heterogeneous embedding components are beneficial to improving forecasting performance. With the increase of d_m, Cospectrumer and its variants are equipped with a more powerful representation and fitting capability

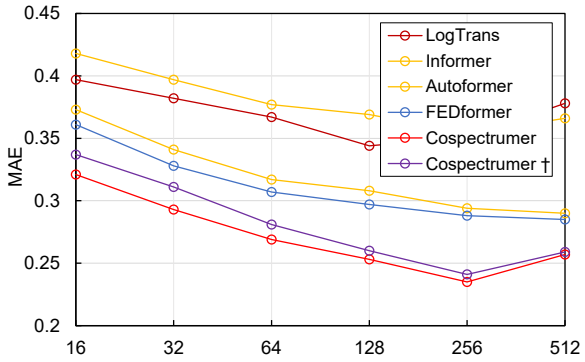
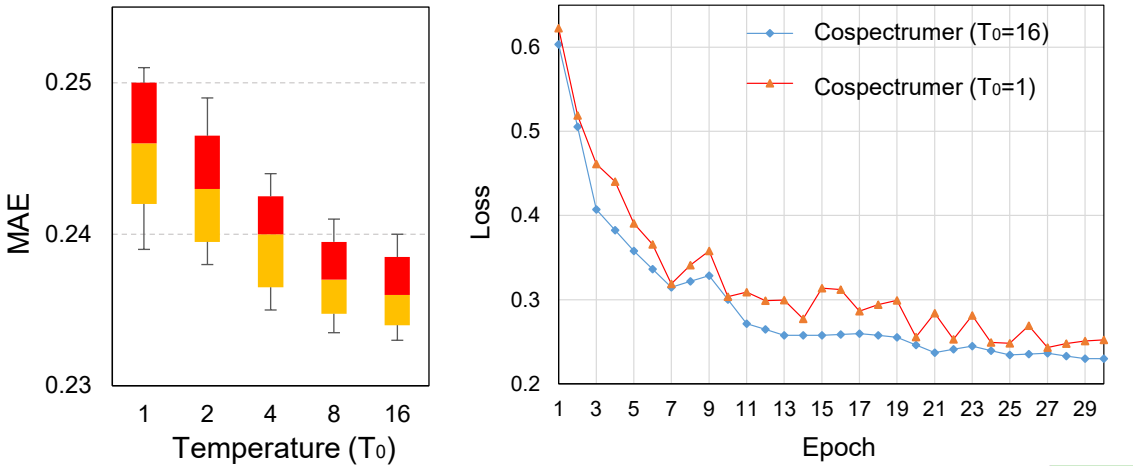


Figure 6.6: Performance comparison of Transformer-based methods with different embedding dimensions on Electricity.

and achieve better performance. When d_m is larger than 256, the performance of the comparative methods degrades, which may result in the overfitting issues.

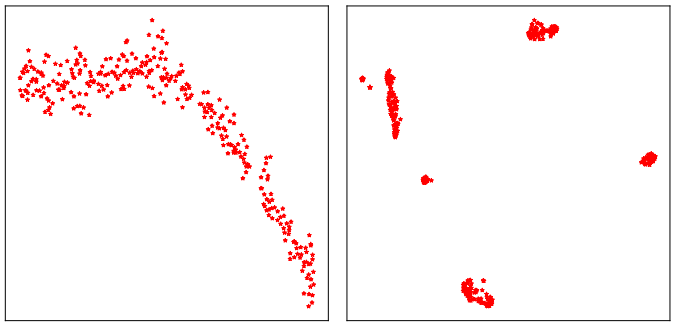


(a) MAE comparison

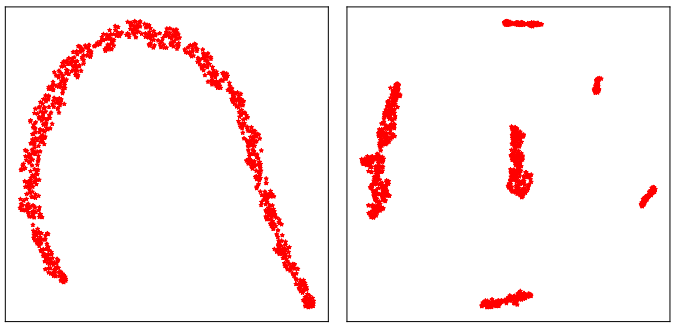
(b) Training convergence

Figure 6.7: (a) MAE comparison under different initial temperatures $T_0 \in \{1,2,4,8,16\}$; (b) training convergence curves of Cospectrumer under $T_0 = 1$ and $T_0 = 16$.

We investigate the impact of temperature softmax on Cospectrumer on Electricity. Specifically, we first compare Cospectrumer under different initial temperatures and report ten-round MAE results using a box plot, as shown in Figure 6.7(a). The results show that, as T_0 increases from 1 to 16, Cospectrumer achieves better performance and is more robust, indicating the necessity and effectiveness of the temperature softmax. Note that we also investigate higher initial temperatures, i.e., $T_0 > 16$, which slows down the convergence and degrades the forecasting performance due to the longer temperature



(a) Electricity (initial) (b) Electricity (termination)



(c) Traffic (initial) (d) Traffic (termination)

Figure 6.8: Clustering visualization using t-SNE on Electricity and Traffic.

decay periods. In addition, we compare the training convergence of Cospectrumer under temperature $T_0 = 16$ and $T_0 = 1$. Figure 6.7(b) shows the curves of the training loss under 30 training epochs. It shows that Cospectrumer with $T_0 = 16$ has a fast convergence speed and converges to a lower MSE loss than that under $T_0 = 1$. The results show that utilizing the temperature softmax to flatten the output of the clustering network can guarantee the stable update of parameters in the forecasting network and benefit the model training. Cospectrumer with $T_0 = 16$ shows a more smooth convergence curve, which benefits from the Gaussian decay function to guarantee a steady update of the temperature. Overall, the results demonstrate that the temperature softmax avoids drastic changes to the forecasting network, and the Gaussian decay function further ensures a stable temperature cooling process.

Finally, we investigate the robustness of Cospectrumer under different learning weights λ on Electricity. Intuitively, since the indicator matrix $\mathbf{C}$ is generated according to the Lipschitz continuity, it guarantees the K-means objective, i.e., Equation (6.3), to a certain extent. The K-means objective can be simplified to impose a constraint on $\mathbf{C}$, precisely $\mathbf{C}^T\mathbf{C} = \mathbf{I}_k$, which can be easily achieved by the softmax activation function.

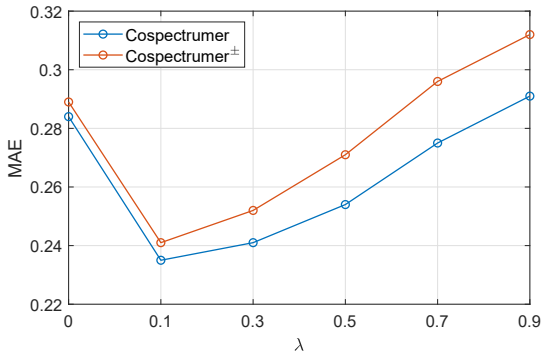


Figure 6.9: Sensitivity analysis on the weight for K-means loss.

Figure 6.9 shows that Cospectrumer and its variants Cospectrumer‡ achieve better performance when λ is relatively small at $\lambda = 0.1$. The results confirm the above analysis that it is not necessary to assign a high value to λ, and a relatively smaller value for λ not only guarantees the K-means objective, it also avoids early convergence of the clustering network.

6.6 Clustering Visualization

To analyze the clustering results and explore the impact of clustering on forecasting, we conduct experiments for $L = 96$, $D = 24$ and $L_{token} = 48$ and visualize their clustering indicator matrix $\mathbf{C}$.

Figure 6.8 shows the clustering visualization of the initial training stage and termination training stage using t-SNE on Electricity and Traffic with the number of clusters $k = 8$. Comparing the two pair figures, their outlines of point distribution are similar, and MTS variables clearly gather into different clusters after the model converges, indicating that the proposed clustering network is able to effectively cluster time series and preserve the distances between time series. In addition, we find five clusters and six clusters shown in Figure 6.8(a) and (b) respectively. The results confirm our claim that the proposed clustering network is able to learn a proper number of clusters when the value k is relatively larger (k is the maximum number of clusters).

Figure 6.10 visualizes how the clustering indicator matrix changes along with the training convergence on Weather using a heatmap. In each matrix, orange and green denote values approximating 1 and 0 respectively. We observe that the matrix approaches a binary matrix when the forecasting network converges, indicating the spectral clustering network also converges to a proper status optimal for the forecasting objective. The result

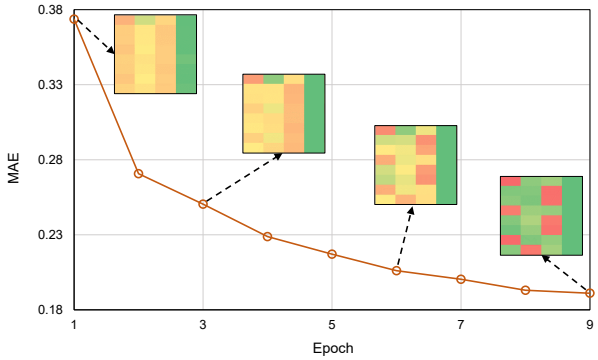


Figure 6.10: Visualization of the change of the clustering indicator matrix on Weather.

shows that the clustering and forecasting networks help each other to synergistically reach convergence, which demonstrates that Cospectrumer effectively captures the common and intrinsic information shared by the two tasks. The results inspire us to utilize the forecasting objective to enhance time series clustering.

6.7 Conclusions

In this chapter, we aim to model inter-/intra-TS couplings and inter-TS heterogeneity to enhance MTS forecasting performance. The main contributions are summarized below:

- A novel clustering-enhanced MTS forecasting model Cospectrumer makes the first attempt to adaptively learn time series clustering for improved MTS forecasting. It achieves this by integrating spectral analysis with Transformer.
- A spectral clustering network effectively learns to segregate target-irrelevant and -relevant heterogeneous TS, which has theoretical guarantees of Lipschitz continuity.
- We propose k-channel Transformers with a cluster-aware attention mechanism to co-attentively learn intra- and inter-TS couplings.
- A heterogeneous embedding module effectively represents the mixed inputs of time series variables, timestamp, and position and captures their second-order couplings.

The extensive experiments on four distinct real-world MTS datasets demonstrate the effectiveness and superiority of Cospectrumer against the state-of-the-art (SOTA) MTS

forecasting baselines. Further, we conduct an ablation study and clustering visualization. They demonstrate the contribution of time series clustering to MTS forecasting and the effectiveness of the proposed network architecture in learning the couplings and heterogeneities of non-IID MTS.

CHAPTER 7

DEEP COUPLING NETWORK FOR MULTIVARIATE TIME SERIES FORECASTING

7.1 Introduction

Previous work on MTS forecasting can be divided into two main approaches: sequential models and GNN-based models. Sequential models, including recurrent neural network (RNN) [55, 170], convolutional neural network (CNN) [14], temporal convolutional network (TCN) [7] and attention mechanisms [54], have been introduced to tackle MTS problems and achieved good performance, which is attributed to their capability of extracting nonlinear intra-series correlations in MTS. However, these models fall short in capturing the inter-series relationships. Recently, GNN-based models [17, 247] have demonstrated promising performance on MTS forecasting with their essential capability to capture the complex inter-series relationships between variables. They mainly adopt a graph neural network to extract the inter-series dependencies and a temporal network to capture the intra-series dependencies.

Unfortunately, the above methods still have some limitations for modeling intra- and inter-dependencies among time series. In Figure 7.1, we show (a) RNN-based models connect two adjacent time step values and ignore the inter-series dependencies. (b) Attention-based models connect different time step values of the variable directly but they also ignore the inter-series dependencies. (c) GNN-based models construct a graph to model the inter-dependencies at each timestamp and then connect the adjacent time

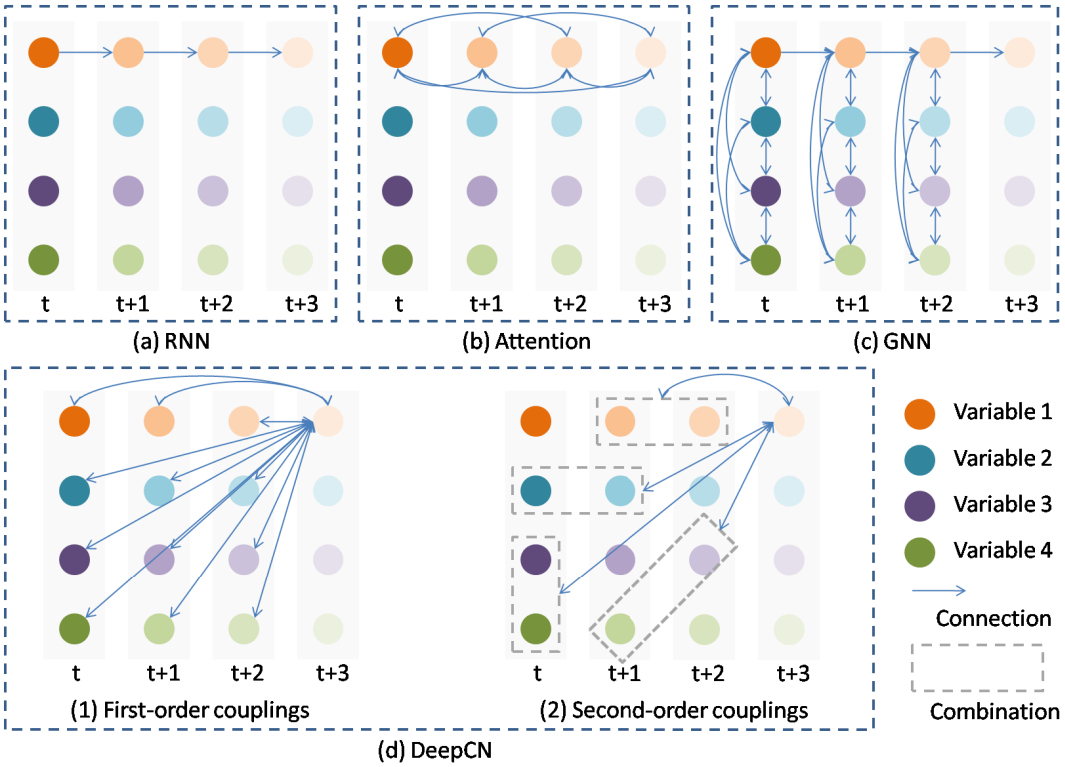


Figure 7.1: Illustration about different models for modeling the intra- and inter-series correlations.

step values for each variable. (d) Our model DeepCN proposes a comprehensive model for intra- and inter-series dependencies which is based on the multi-order couplings of different time lag. The maximum length of the signal traversing path of RNN-based models is $\mathcal{O}(L)$, thus making them difficult to learn intra-series dependencies between distant positions [132]. Attention-based models [104, 243, 289] shorten the maximum path to be $\mathcal{O}(1)$ but they ignore the inter-series dependencies. Although most existing state-of-the-art GNN-based models [6, 247, 248, 264] for MTS forecasting construct a graph to model the inter-series dependencies, they rely on LSTM or GRU to capture the intra-series dependencies which encounter the same limitations as the RNN-based models in modeling the intra-series dependencies. Besides, they can not directly connect the different variables at different timestamps, which leads them to not obtain the abundant inter-series dependencies.

Moreover, in practice the multivariate relationships between time series exhibit various dependencies [235]. Accordingly, how to model the complex intra- and inter-series dependencies calls for a comprehensive analysis. Couplings [139], referring to any relationship or interaction which connects two or more variables, include diverse correlations,

dependency, interactions and hierarchy. Learning such couplings [139] can explore more comprehensive and stronger representations by revealing and embedding various couplings on complex data [293], and has achieved good results in many domains, including recommendation [90, 276], image source identification [85] and financial market analysis [25]. Motivated by this, we capture the complex intra- and inter-series dependencies by exploring the multi-order couplings as shown in Figure 7.1(d).

In light of the above discussion, in this section, we theoretically analyze the relationship modeling among time series and propose a novel deep coupling network (named DeepCN) for multivariate time series forecasting. Concretely, first we revisit the relationships among time series from the perspective of mutual information, and then based on the analysis, we propose a coupling mechanism to comprehensively model the intra- and inter-series correlations among time series explicitly via exploring the diverse and hierarchical couplings on MTS data. Subsequently, we leverage a coupled variable representation module to encode the variable relationship representations since different variables exhibit different patterns. Finally, we use an inference module to make predictions by one forward step which can avoid error accumulation. We compare the proposed DeepCN with state-of-the-art baselines on five real world datasets, where experimental results show the superiority of DeepCN and demonstrate the effectiveness of the proposed neural architecture and coupling mechanism in DeepCN.

7.2 Problem Formulation

We are given a multivariate time series input $X \in \mathbb{R}^{N \times T}$ where N is the number of time series and T is the length of timestamps. If $Z \in \mathbb{R}^{1 \times T}$ belongs to X and is the target variable, given the historical T observations X, we predict the next τ timestamps $\hat{Z}$, and the deterministic forecasting task can be formulated as follows:

$$\hat{Z}^{T+1:T+\tau} = F\left(Z^{1:T}, X^{1:T}; \Theta\right) \tag{7.1}$$

where F is the prediction model and Θ is the parameters of F. For multivariate time series forecasting, it is important to exploit the intra-series relationship between $Z^{1:T}$ and the inter-series relationship between $Z^{1:T}$ and $X^{1:T}$.

7.3 Coupling Analysis

As aforementioned introduction, the intra- and inter-dependencies among time series are complex. In this section, we revisit the relationships among time series from the perspective of mutual information.

According to Equation 3.19, we enumerate the subset of S and the right side of the equation can be expanded as follows:

$$\begin{aligned}\sum_{s\subseteq S} I(\{s\cup Z\}) = &\sum_{i=1}^{N} I(X_i;Z) + \sum_{i=1}^{N}\sum_{j=i+1}^{N} I(\{X_i,X_j,Z\}) \\ &+ \sum_{i=1}^{N}\sum_{j=i+1}^{N}\sum_{k=j+1}^{N} I(\{X_i,X_j,X_k,Z\}) + ...\end{aligned} \tag{7.2}$$

where $I(X_i;Z)$ represents the relationships between X_i and Z, $I(\{X_i,X_j,Z\})$ represents the relationships between Z and $\{X_i,X_j\}$, and $I(\{X_i,X_j,X_k,Z\})$ represents the relationships between Z and $\{X_i,X_j,X_k\}$. The Equation 7.2 demonstrates that the relationships between Z and $X_{1:N}$ are the summation of mutual information between different combination of $X_{1:N}$ and Z. We denote this relationship as multi-order couplings.

Furthermore, the time lag effect between time-series variables is a common phenomenon in real-world MTS scenarios, for example, the time lag influence between two financial assets (e.g. dollar and gold) of a portfolio. In other words, X^t may be influenced by $X^{t-1},X^{t-2},...,X^{t-T}$. Due to time delay in time series data, the relationship between variables are more complicated. Accordingly, to model the relationship between X_1 and X_2 at time t, we not only need to consider data relation at time t but also at time $t-1,...,t-T$.

Combined with the above introduction (i.e., multi-order couplings and time lag effect), the relationship between variable Z and $X_{1:N}$ not only needs to consider their relationship at time t, but also consider their time lag effect at time $t-1$, ..., $t-T$.

Accordingly, we redefine the relationship as follows:

(7.3)

$$
\begin{aligned}
I(Z;X_1,X_2,...,X_N) &= I(Z;X_1^t,X_1^{t-1},...,X_1^{t-T},X_2^t,X_2^{t-1},...,X_2^{t-T},...,X_N^t,X_N^{t-1},...,X_N^{t-T}) \\
&= \underbrace{\sum_{i=1}^{N} I(X_i^t;Z) + \sum_{i=1}^{N} I(X_i^{t-1};Z) + ... + \sum_{i=1}^{N} I(X_i^{t-T};Z)}_{first-order \quad coupling} \\
&+ \underbrace{\sum_{i=1}^{N}\sum_{j=i+1}^{N}\sum_{l=0}^{T}\sum_{m=0}^{T} I(\{X_i^{t-l},X_j^{t-m},Z\})}_{second-order \quad coupling} \\
&+ \underbrace{\sum_{i=1}^{N}\sum_{j=i+1}^{N}\sum_{k=j+1}^{N}\sum_{l=0}^{T}\sum_{m=0}^{T}\sum_{n=0}^{T} I(\{X_i^{t-l},X_j^{t-m},X_k^{t-n},Z\}) + ...}_{third-order \quad coupling}
\end{aligned}
$$

where T is the delay time step. Similar to Equation 7.2, the right part of the Equation 7.3 is expanded as the summation of mutual information between different combinations of X and Z. From Equation 7.3, we can conclude that the relationship between Z and X can be modeled through multi-order couplings between different time lag of X and Z. When the target variable Z is one of $X_{1:N}$, Equation 7.3 contains both intra-series (e.g., I(X_i^t;X_i^t) when Z=X_i) and inter-series relationship. Therefore, from the mutual information perspective, the intra- and inter-dependencies among time series can be modeled by multi-order couplings of different time lag of X.

As shown in Figure 7.1, RNN-based models and Attention-based models mainly account for intra-series relationship. Corresponding to Equation 7.3, they can not capture the couplings between X_i and X_j ($i \neq j$). Besides, RNN-based models can not explicitly attend to time lag effect while Attention-based models can do because they can connect directly between different time lags. That is to say, Attention-based models can directly capture the couplings between X_i^{t-l} and X_i^{t-m} ($l \neq m \pm 1$) while RNN-based models can not. And this is the reason that Attention-based models perform better than RNN-based models in most cases. GNN-based models can model both intra-series and inter-series relationship. However, they also can not explicitly attend to time lag effect of both intra- and inter-series. In other words, GNN-based models can not directly account for the couplings between X_i^{t-l} and X_j^{t-m} ($l \neq m \pm 1, i \neq j$). This limits the capability of GNN for MTS forecasting and sometimes lead them to achieve worse accuracy than Attention-based models. Accordingly, in this section, based on the Equation 7.3, we design a comprehensive model for intra- and inter-series relationships among time series explicitly by exploring the diverse and hierarchical couplings in 7.4.2.

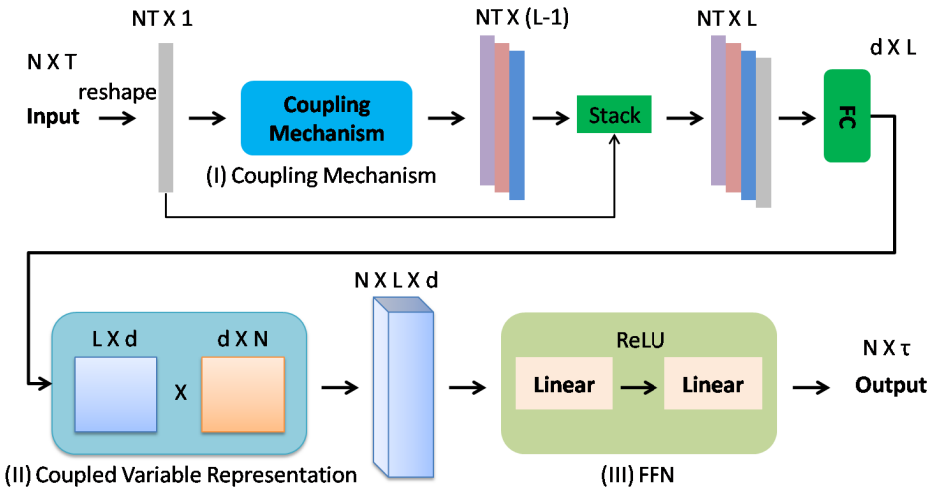


Figure 7.2: The overview framework of DeepCN.

In the experimental part, we verify that the relationship between variables has the time lag effect through the parameter sensitivity test about input length. Meanwhile, we analyze the multi-order coupling characteristics of relationships through the coupling mechanism study. The details are introduced in section 7.5.5.

7.4 Deep Coupling Network For Multivariate Time Series Forecasting

In this section, we introduce a novel deep coupling network named DeepCN for MTS forecasting, which comprehensively models the intra- and inter-dependencies among time series according to the analysis in Section 7.3.

7.4.1 Overview Framework

The overall framework of DeepCN is illustrated in Figure 7.2. Given a multivariate time series input $X \in \mathbb{R}^{N \times T}$ where N is the number of variables and T is the length of timestamps, first we conduct a reshape operator to transform the input $X \in \mathbb{R}^{N \times T}$ to a vector $V \in \mathbb{R}^{1 \times NT}$. Next, V serves as input to the coupling mechanism which is designed to comprehensively explore the complicated intra- and inter-dependencies among time series according to the analysis in Section 7.3. It applies explicit variable crossing of different time lags and outputs the couplings $C \in \mathbb{R}^{(L-1) \times NT}$ where L is the number of

orders (details see in Subsection 7.4.2). Then we stack C with V and get the multi-order couplings among time series $C' \in \mathbb{R}^{L \times NT}$.

After that, C' is fed to a fully-connected neural network and output the dense representation $h \in \mathbb{R}^{L \times d}$ where d is the dimension size. Since different variables exhibit different patterns, we initialize a variable embedding matrix $W \in \mathbb{R}^{N \times d}$ to embed the variable and perform a multiplication $h \times W^T$ to represent variable relationships $O \in \mathbb{R}^{N \times L \times d}$. Finally, we utilize the FFN composed of feed forward networks and activation function to predict the next τ timestamps $\hat{X} \in \mathbb{R}^{N \times \tau}$ by one forward step. The above content is explained in detail below.

7.4.2 Coupling Mechanism

One key component in our proposed model is the coupling mechanism which explicitly explores diverse and hierarchical couplings to capture the complicated intra- and inter-dependencies among time series. In Section 7.3, we have analyzed the characteristics of relationships between variables. In this subsection, we design a coupling mechanism to represent relationships based on those characteristics (i.e., multi-order couplings and time lag effect).

According to Equation 7.3, the relationships among time series can be expressed through the couplings between different combinations of time lags of time series. Intuitively, the multi-order couplings can be modeled by cross feature [227]. In light of the content introduced in Subsection 2.1.1, we understand that combinatorial features are essential in commercial models [121]. Cartesian Product model, as a state-of-the-art instance of combinatorial based model [288], is an explicit model. Motivated by this, to explore the L-order couplings between Z and $X_{1:N}$, we leverage the Cartesian product to illustrate how to calculate it. The Cartesian product $Cart_L$ can be defined as follows:

$$Cart_L = \{(\underbrace{x_i, x_j, ..., x_l}_{L}) | x_{i,j,...,l} \in X_i^{1:T}, \forall i, j, ..., l = 1, 2, ..., N\} \tag{7.4}$$

where T is the length of timestamps. To illustrate this definition more intuitively, we use second-order couplings, five-time delay steps, and four variables as an example in Figure 7.3. In the figure, we take four variables, five time lags and second-order as an example. From the figure, there are various combinations from cross-variable and cross-time, and several of them are marked with dotted lines on the figure. Moreover, the combinations include both intra-series (e.g., $\{X_1^{t-2} X_1^{t-3}\}, \{X_1^{t-1} X_1^{t-3}\}$) and inter-series (e.g., $\{X_1^t X_3^t\}, \{X_2^{t-2} X_4^{t-1}\}$) information. In the example, there are various combinations

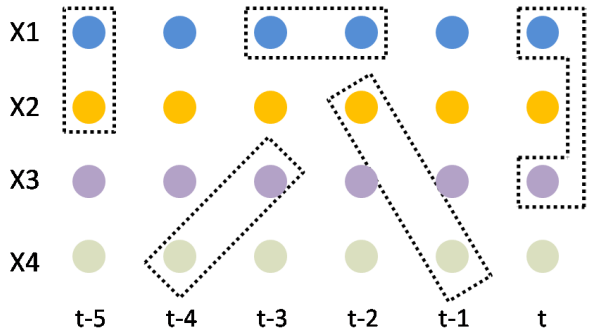


Figure 7.3: Multi-order couplings diagram in the deep coupling network.

from cross variable and cross time, and several of them are marked with dotted lines in the figure.

From Equation 7.4, we can learn that $Cart_L$ takes on all possible combinations of values from 1 to T. In addition, the combinations contain both intra-series (e.g., $X_1^t X_1^{t-1} ... X_1^{t-T}$) and inter-series (e.g., $X_1^t X_2^t ... X_L^t$) relationships. Inspired by the cross network [227] which has linear complexity, we express these combinations explicitly via efficiently learning all types of cross features. First, we convert the input matrix $X \in \mathbb{R}^{N \times T}$ to a vector $V \in \mathbb{R}^{NT}$ by a reshape operation. Then we adopt the method of feature interactions to model the relationships, and we can get the L-order couplings as follows:

$$C^L = V(C^{L-1})^T W_{L-1} + b_{L-1}, L > 1$$
$$C^1 = V, L = 1 \tag{7.5}$$

where $W_{L-1} \in \mathbb{R}^{NT}, b_{L-1} \in \mathbb{R}^{NT}$ are weight parameters and bias parameters, respectively. It should be noted that the first-order coupling, namely C^1, is the input V itself. Detailed calculation process is shown in Figure 7.4 and the algorithm is described in Algorithm 2. In the figure, we first transform the input matrix $X = \mathbb{R}^{N \times T}$ to an input vector $V = \mathbb{R}^{NT}$ by a reshape operation where N is the number of variables and T is the length of timestamps. Then we calculate the different order couplings respectively. V^T is the first-order. According to Equation 7.5, we can find that it can fully explore all types of combinations of X. Different values of L corresponds to different order couplings. For example, when $L = 2$, C^L corresponds to second-order couplings.

Then we can express the hierarchical couplings C which are composed of various order couplings as follows:

$$C = (C^1, C^2, ..., C^L) \tag{7.6}$$

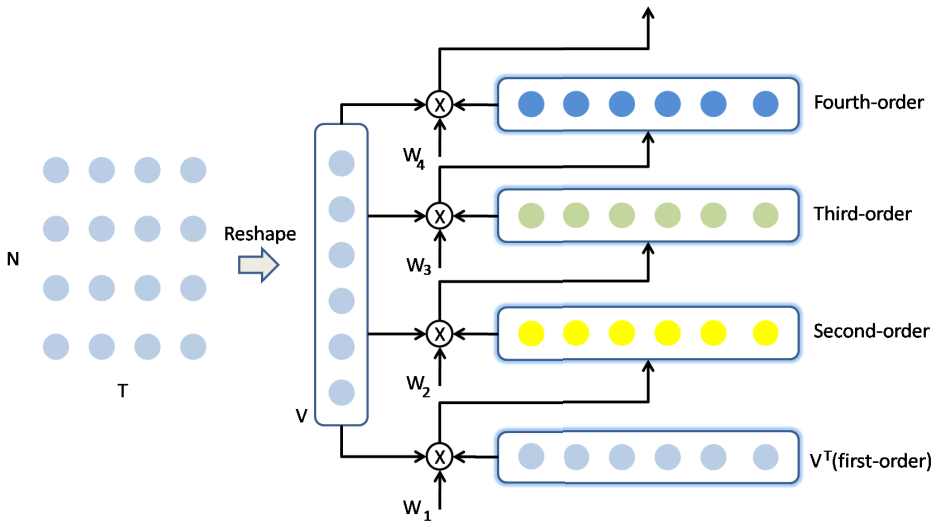


Figure 7.4: Coupling-based model for relationships between variables.

where L is the total number of orders.

Complexity Analysis. Let L denotes the total number of orders, N denotes the number of variables, and T denotes time delay steps. Then the total number of parameters, namely W and b in Equation 7.5, is

$$L \times N \times T \times 2. \tag{7.7}$$

From formula 7.7, we can conclude that the complexity of the coupling mechanism is linear and is linearly proportional to the orders L, the number of variables N, and the time delay steps T. In general, T is much smaller than N.

7.4.3 Coupled Variable Representation Module

The output C of coupling mechanism represents comprehensive intra- and inter-dependencies among time series (i.e., first-order coupling, second-order coupling, and so on). Since different variables exhibit different patterns, we leverage a coupled variable representation module to learn the variable relationship representation.

First, the output $C \in \mathbb{R}^{L \times NT}$ of coupling mechanism is fed to a fully-connected network and output the dense representation $h \in \mathbb{R}^{L \times d}$ where d is the dimension size. Then we initialize a variable embedding weight matrix $W \in \mathbb{R}^{N \times d}$, and perform multiplication with the dense representation h and output $O \in \mathbb{R}^{N \times L \times d}$. The calculation can be described

Algorithm 2 Coupling Mechanism

Input: multivariate time series $X \in \mathbb{R}^{N \times T}$
Output: the multi-order couplings $C \in \mathbb{R}^{L \times NT}$
Given the multivariate time series input $X \in \mathbb{R}^{N \times T}$
Reshape X into a vector $V \in \mathbb{R}^{NT \times 1}$
Initialize the couplings C
$V_0 = V$
$C[0] = V$
for $l = 1, ..., L$ **do**
    Initialize weight matrix $W \in \mathbb{R}^{NT \times NT}$
    Initialize bias vector $b \in \mathbb{R}^{NT \times 1}$
    $V_w = V_{l-1} \times W$
    $V_l = V_w \times V_0 + b$
    $C[l] = V_l$
end for
return C

as follows:

$$\begin{aligned} h &= CW_h + b_h \\ O &= W \times h^T \end{aligned} \tag{7.8}$$

where $W_h \in \mathbb{R}^{NT \times d}$ and $b_h \in \mathbb{R}^{L \times d}$ is the weight matrix and bias matrix, respectively.

7.4.4 Inference Module

The inference module, composed of ReLU and the fully-connected network, is used to make predictions according to the output of coupled variable representation O. To avoid accumulations of errors, we output all predicted results by one forward step which turns out to be more efficient and stable than step-by-step prediction in Section 7.5. The inference module is formulated as follows:

$$\hat{X} = \sigma(OW_1 + b_1)W_2 + b_2 \tag{7.9}$$

where $O \in \mathbb{R}^{N \times L \times d}$ is the output of coupled variable representation module, σ is the activation function, $W_1 \in \mathbb{R}^{(L*d) \times d_h}, W_2 \in \mathbb{R}^{d_h \times \tau}$ are the weights, and $b_1 \in \mathbb{R}^{d_h}$, $b_2 \in \mathbb{R}^{\tau}$ are the biases, and d_h is the inner-layer dimension size.

The final loss can be formulated as follows:

$$\mathscr{L}(\hat{X}, X; \Theta) = \sum_t \left\| \hat{X}^{t+1:t+\tau} - X^{t+1:t+\tau} \right\|_2^2 \tag{7.10}$$

where τ is the predicted length, Θ is the learnable parameters, $\hat{X}$ is the prediction values and X is the ground truth.

7.5 Experiments

In this section, we perform extensive experiments to evaluate the prediction accuracy and efficiency of our proposed model DeepCN, and provide analyses about DeepCN, including the study of coupling mechanism, ablation study, and parameter analysis.

7.5.1 Datasets

We empirically perform experiments on five real-world datasets, including traffic, energy, web traffic, electrocardiogram, and COVID-19. These datasets are summarized in Table 7.1. All datasets are normalized using the min-max normalization. Except the COVID-19 dataset, we split the other datasets into training, validation, and test sets with the ratio of 7:2:1 in a chronological order. For the COVID-19 dataset, the ratio is 6:2:2.

Solar[1]: This data set is about solar power collected by NREL (National Renewable Energy Laboratory). We use the usage of a state as the data set which contains 593 points. The data has been collected from 2006/01/01 to 2016/12/31 with the sampling interval of every 1 hour.

Wiki[2]: The data set contains a number of daily views of different Wikipedia articles which have been collected from 2015/7/1 to 2016/12/31. It consists of approximately 145k time series and we choose 2k of them as our experimental data set due to limited computing resources.

Traffic[3]: The data set contains a number of hourly traffic data of 963 San Francisco freeway car lanes which are collected from 2015/01/01 with the sampling interval of every 1 hour.

ECG[4]: The data set is about Electrocardiogram(ECG) from the UCR time-series classification archive [45]. It contains 140 nodes and each node has a length of 5000.

COVID-19[5]: The dataset is about COVID-19 hospitalization in the U.S. state of California (CA) from 01/02/2020 to 31/12/2020 provided by the Johns Hopkins University with the sampling interval of every day.

[1] https://www.nrel.gov/grid/solar-power-data.html
[2] https://www.kaggle.com/c/web-traffic-time-series-forecasting/data
[3] https://archive.ics.uci.edu/ml/datasets/PEMS-SF
[4] http://www.timeseriesclassification.com/description.php?Dataset=ECG5000
[5] https://github.com/CSSEGISandData/COVID-19

Table 7.1: Summary of Experimental Datasets

Datasets	Solar	Wiki	Traffic	ECG	COVID-19
Samples	3650	803	10560	5000	335
Variables	592	5000	963	140	55
Granularity	1hour	1day	1hour	-	1day
Start time	01/01/2006	01/07/2015	01/01/2015	-	01/02/2020

7.5.2 Baselines

We compare our proposed model with the following representative and SOTA models, including the classic model VAR [237], DNN-based models, matrix factorization models, GNN-based models, and Transformer-based models.

Classic Model.

- **VAR**[6]: VAR [237] is a classic linear autoregressive model. We use Statsmodels library which is a python package that provides statistical computations to realize the VAR.

DNN-based models.

- **LSTNet**[7]: LSTNet [111] uses a CNN to capture inter-variable relationships and an RNN to discover long-term patterns. In our experiment, we use the settings where the number of CNN hidden units is 100, the kernel size of the CNN layers is 4, the dropout is 0.2, the RNN hidden units is 100, the number of RNN hidden layers is 1, the learning rate is 0.001 and the optimizer is Adam.

- **TCN**[8]: TCN [7] is a causal convolution model for regression prediction. We utilize the same configuration as the polyphonic music task exampled in the open source code where the dropout is 0.25, the kernel size is 5, the hidden units is 150, the number of levels is 4 and the optimizer is Adam.

- **SFM**[9]: On the basis of the LSTM model, SFM [274] introduces a series of different frequency components in the cell states. We use the default settings as the authors recommended where the learning rate is 0.01, the frequency dimension is 10, the hidden dimension is 10 and the optimizer is RMSProp.

[6] `https://www.statsmodels.org`
[7] `https://github.com/laiguokun/LSTNet`
[8] `https://github.com/locuslab/TCN`
[9] `https://github.com/z331565360/State-Frequency-Memory-stock-prediction`

Matrix factorization models.

- **DeepGLO**[10]: DeepGLO [188] models the relationships among variables by matrix factorization and employs a temporal convolution neural network to introduce non-linear relationships. We use the default setting as our experimental settings for wiki, electricity and traffic datasets. For covid datasets, the vertical and horizontal batch size is set to 64, the rank of the global model is set to 64, the number of channels is set to [32, 32, 32, 1], and the period is set to 7.

GNN-based models.

- **StemGNN**[11]: StemGNN [17] leverages GFT and DFT to capture dependencies among variables in the frequency domain. We use the default setting of stemGNN as our experiment setting where the optimizer is RMSProp, the learning rate is 0.0001, the stacked layers is 5, and the dropout rate is 0.5.

- **MTGNN**[12]: MTGNN [247] proposes an effective method to exploit the inherent dependency relationships among multiple time series. We download the source code from: `https://github.com/nnzhan/MTGNN`. Because the experimental datasets have no static features, we set the parameter load_static_feature to false. We construct the graph by the adaptive adjacency matrix and add the graph convolution layer. Regarding other parameters, we adopt the default settings.

- **GraphWaveNet**[13]: GraphWaveNet [248] introduces an adaptive dependency matrix learning to capture the hidden spatial dependency. Since our datasets have no prior defined graph structures, we use only adaptive adjacent matrix. We add a graph convolutional layer and randomly initialize the adjacent matrix. We adopt the default setting as our experimental settings where the learning rate is 0.001, the dropout is 0.3, the number of epoch is 50, and the optimizer is Adam.

- **AGCRN**[14]: AGCRN [6] proposes a data-adaptive graph generation module for discovering spatial correlations from data. We use the default settings as our experimental settings where the embedding dimension is 10, learning rate is 0.003, and the optimizer is Adam.

[10] `https://github.com/rajatsen91/deepglo`
[11] `https://github.com/microsoft/StemGNN`
[12] `https://github.com/nnzhan/MTGNN`
[13] `https://github.com/nnzhan/Graph-WaveNet`
[14] `https://github.com/LeiBAI/AGCRN`

Transformer-based models.

- **Informer**[15]: Informer [289] leverages an efficient self-attention mechanism to encode the dependencies among variables. We use the default settings as our experimental settings where the dropout is 0.05, the number of encoder layers is 2, the number of decoder layers is 1, the learning rate is 0.0001, and the optimizer is Adam.

- **Reformer**[16]: Reformer [104] combines the modeling capacity of a Transformer with an architecture that can be executed efficiently on long sequences and with small memory use. We use the recommended settings as the experimental settings.

- **Autoformer**[17]: Autoformer [243] proposes a decomposition architecture by embedding the series decomposition block as an inner operator, which can progressively aggregate the long-term trend part from intermediate prediction. We use the recommended settings as our experimental settings with 2 encoder layers and 1 decoder layer.

7.5.3 Experimental Setup

We perform our experiments with the hardware environment of one NVIDIA RTX 3080 card. Our code is implemented by Python 3.6 with PyTorch 1.9. Our model is optimized with RMSprop optimizer and the learning rate is 0.00001. We normalize the input by the min-max way. For all datasets, batch size is set to 32. The number of epoch is 50. For traffic and COVID-19 datasets, the dimension size d is set to 512. For Wiki, Solar and ECG datasets, the dimension size d is set to 1024. For COVID-19 and ECG datasets, the number of orders L is set to 2. For Traffic and Wiki dataset, L is set to 4. For Solar dataset, L is set to 3. In the inference module, the hidden size is 1024 and the activation function is $ReLU$. We use MAE and RMSE as metrics.

7.5.4 Results

Single-step forecasting. We compare our model DeepCN with the other twelve models on five different datasets with the input length being 12 and the prediction length being 12. The results are summarized in Table 7.2. As can be seen from the result table,

[15] https://github.com/zhouhaoyi/Informer2020
[16] https://github.com/thuml/Autoformer
[17] https://github.com/thuml/Autoformer

Dataset	Solar		Wiki		Traffic		ECG		COVID-19	
Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
VAR	0.184	0.234	0.057	0.094	0.535	1.133	0.120	0.170	0.226	0.326
SFM	0.161	0.283	0.081	0.156	0.029	0.044	0.095	0.135	0.205	0.308
LSTNet	0.148	0.200	0.054	0.090	0.026	0.057	0.079	0.115	0.248	0.305
TCN	0.176	0.222	0.094	0.142	0.052	0.067	0.078	0.107	0.317	0.354
DeepGLO	0.178	0.400	0.110	0.113	0.025	0.037	0.110	0.163	0.191	0.253
Reformer	0.234	0.292	0.047	0.083	0.029	0.042	0.062	0.090	0.182	0.249
Informer	0.151	0.199	0.051	0.086	0.020	0.033	0.056	0.085	0.200	0.259
Autoformer	0.150	0.193	0.069	0.103	0.029	0.043	0.055	0.081	0.189	0.241
GraphWaveNet	0.183	0.238	0.061	0.105	0.013	0.034	0.093	0.142	0.201	0.255
StemGNN	0.176	0.222	0.190	0.255	0.080	0.135	0.100	0.130	0.421	0.508
MTGNN	0.151	0.207	0.101	0.140	0.013	0.030	0.096	0.145	0.394	0.488
AGCRN	0.143	0.214	0.044	0.079	0.084	0.166	0.055	0.080	0.254	0.309
DeepCN (ours)	**0.132**	**0.172**	**0.042**	**0.078**	**0.011**	**0.022**	**0.051**	**0.076**	**0.170**	**0.232**
Improvement	10.8%	10.9%	4.5%	1.3%	15.4%	26.7%	7.3%	8.4%	6.6%	3.7%

Table 7.2: Single step forecasting error results (MAE and RMSE) of DeepCN and other baseline models on five datasets with the prediction length being 12.

DeepCN achieves good performances on all datasets. On the Solar dataset, DeepCN improves 10.8% on MAE and 10.9% on RMSE. On the Wiki dataset, it improves 4.5% on MAE and 1.3% on RMSE. On the ECG dataset, it improves 7.3% on MAE and 8.4% on RMSE. On the COVID-19 dataset, it improves 6.6% on MAE and 3.7% on RMSE. Especially, on the traffic dataset, DeepCN improves 15.4% on MAE and 26.7% on RMSE compared with the best baseline. The reason why DeepCN performs exceptionally well on the traffic data set is because of the strong coupling in the traffic dataset. For example, adjacent nodes affect each other and adjacent areas also affect each other. If a road is jammed, it inevitably affects other roads. We conduct more experiments on the couplings of the traffic dataset and the result is shown in Figure 7.9(a). From the figure, we conclude that there is a multi-order couplings in the traffic dataset. This also explains why DeepCN is more accurate. It is because the multi-order couplings is within our consideration.

Among these baseline models, transformer-based models and GNN-based models have achieved more competitive performances than other DNN-models. As shown in Figure 7.1, DNN-based models can not attend to time lag effect directly and ignore the inter-series dependencies. The two inherent defects affect their ability to capture the dependencies among time series. Moreover, GNN-based models perform well on Solar, Wiki, Traffic and ECG datasets while transformer-based models achieve good results only on COVID-19 and ECG datasets. This is because that compared with transformer-based models, GNN-based models consider the inter-series dependencies. This also demonstrates that inter-series dependencies are valuable for MTS forecasting especially for the tight coupling scenarios such as traffic forecasting.

Horizon	3		6		9		12	
Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
LSTNet	0.085	0.178	0.128	0.202	0.128	0.202	0.128	0.203
DeepGLO	0.083	0.142	0.093	0.161	0.107	0.157	0.110	0.163
Informer	0.055	0.083	0.055	0.084	0.057	0.086	0.056	0.085
GraphWaveNet	0.090	0.139	0.092	0.141	0.095	0.145	0.097	0.149
StemGNN	0.090	0.130	0.100	0.130	0.090	0.129	0.100	0.130
MTGNN	0.092	0.140	0.094	0.140	0.095	0.144	0.096	0.145
DeepCN (ours)	**0.050**	**0.076**	**0.050**	**0.076**	**0.051**	**0.076**	**0.051**	**0.076**
Improvement	9.1%	8.4%	9.1%	9.5%	10.5%	11.6%	8.9%	10.6%

Table 7.3: Multi-step forecasting error comparison (MAE and RMSE) of DeepCN with six baseline models on ECG dataset with the prediction length in $\{3, 6, 9, 12\}$.

Horizon	3		6		9		12	
Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
VAR	0.047	0.076	0.095	0.150	0.182	0.319	0.535	1.133
LSTNet	0.016	0.038	0.019	0.045	0.023	0.051	0.026	0.057
DeepGLO	0.020	0.036	0.022	0.036	0.024	0.038	0.025	0.037
Informer	0.019	0.031	0.020	0.032	0.020	0.032	0.020	0.033
GraphWaveNet	0.011	0.027	0.013	0.031	0.013	0.030	0.013	0.034
StemGNN	0.050	0.093	0.070	0.121	0.090	0.144	0.080	0.135
MTGNN	0.011	0.026	0.012	0.027	0.012	0.028	0.013	0.030
DeepCN (ours)	**0.009**	**0.020**	**0.010**	**0.021**	**0.011**	**0.021**	**0.011**	**0.022**
Improvement	18.2%	23.1%	16.7%	22.2%	8.3%	25.0%	15.4%	26.7%

Table 7.4: Multi-step forecasting error comparison (MAE and RMSE) of DeepCN and seven baseline models on Traffic dataset with the prediction length in $\{3, 6, 9, 12\}$.

Multi-step forecasting. In order to further evaluate the accuracy of DeepCN under different prediction lengths, we conduct more experiments on multi-step forecasting, including 3, 6, 9, and 12 steps. We perform experiments on ECG, traffic and Wiki datasets, respectively. For the ECG dataset, the input length of all models is set to 12. The coupling number L of DeepCN is set to 2. As you can see from the results in Table 7.3, DeepCN achieves good performances in multi-step forecasting tasks as it improves an average of 9.4% on MAE and 10% on RMSE. Among the baselines, Informer performs better than others since it can model the intra-series dependencies directly. For the traffic dataset, the input length of all models is also set to 12. The coupling number L of DeepCN is set to 4. The results are shown in Table 7.4 and we can find

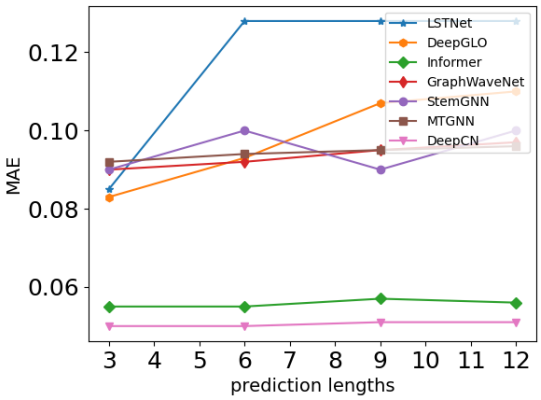
Length	3		6		9		12	
Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
GraphWaveNet	0.061	0.105	0.061	0.105	0.061	0.105	0.061	0.104
StemGNN	0.157	0.236	0.159	0.233	0.232	0.311	0.220	0.306
AGCRN	0.043	0.077	0.044	0.078	0.045	0.079	0.044	0.079
MTGNN	0.102	0.141	0.091	0.133	0.074	0.120	0.101	0.140
Informer	0.053	0.089	0.054	0.090	0.059	0.095	0.059	0.095
DeepCN (ours)	**0.041**	**0.076**	**0.042**	**0.076**	**0.042**	**0.077**	**0.042**	**0.078**
Improvement	4.7%	1.3%	4.5%	2.6%	6.7%	2.5%	4.5%	1.3%

Table 7.5: Multi-step forecasting error comparison (MAE and RMSE) of DeepCN with five baseline models on Wiki dataset with the prediction length in $\{3, 6, 9, 12\}$.

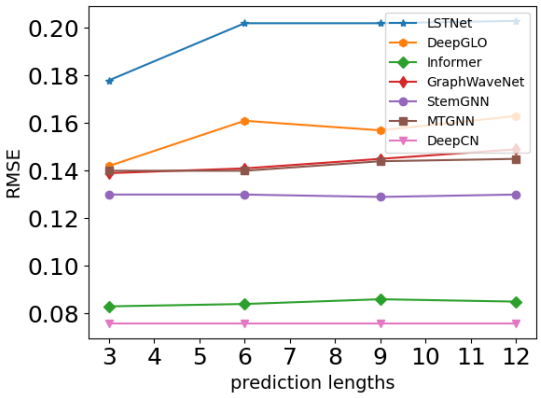
that DeepCN improves an average of 14.7% on MAE and 24.3% on RMSE. Among the baselines, MTGNN shows good performances because it has good capability to capture the inter-series dependencies. For the Wiki dataset, we choose GNN-based models and transformer-based model Informer as the baseline models, the input length is set to 12 and the number of orders L is set to 4. The results in Table 7.5 demonstrate that our proposed model improves an average of 5.1% on MAE and 1.9% on RMSE. Among the baselines, AGCRN achieves competitive results since it constructs the graph structure (i.e., the inter-series dependencies) adaptively from time series data.

Moreover, Figure 7.5 and Figure 7.6 exhibit the changing curve of the accuracy under different steps on the ECG dataset and traffic dataset, respectively. From Figure 7.5, we can find that as the steps increase, the accuracy rate of a classic model (VAR) decreases. It also shows that the accuracy of Informer is closer to ours, but LSTNet's performance is not quite ideal on the ECG dataset. Figure 7.6 shows that the accuracy of GNN-based models is closer to ours, whereas StemGNN presents poor accuracy on the traffic dataset. In addition, from Figure 7.5 and Figure 7.6, we can observe that the accuracy of our proposed model DeepCN is stable when prediction length increases because DeepCN utilizes one forward step to make a prediction which can avoid error accumulations. Informer also makes predictions by one forward step and its accuracy shows stable while StemGNN adopting a rolling strategy has fluctuations in performance.

In Figure 7.7, the three different color curves stand for slices of the ground truth, Informer, and DeepCN, respectively. Four nodes were randomly selected from 140 in the dataset as observation variables to compare the ground truth with prediction values of Informer and DeepCN. The above four subgraphs correspond to a node, respectively. It



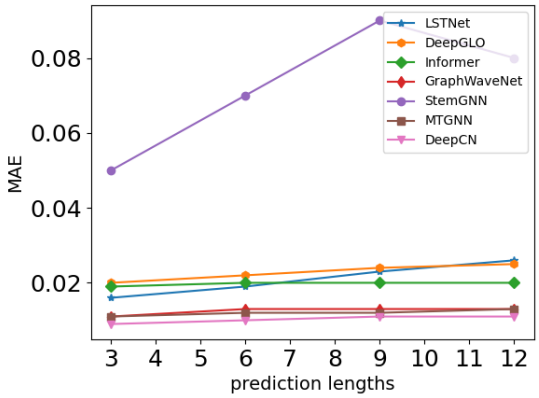
(a) Analysis on MAE



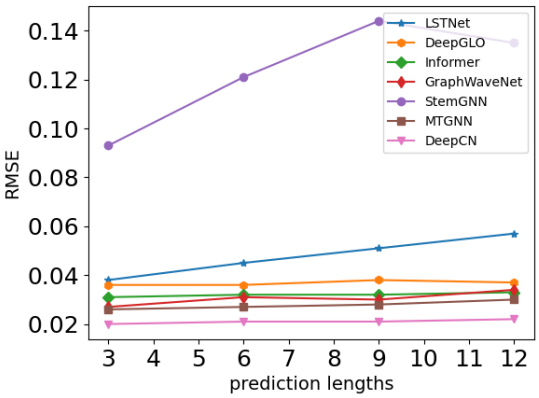
(b) Analysis on RMSE

Figure 7.5: Multi-step forecasting error result analysis (MAE and RMSE) of DeepCN and five baseline models on ECG dataset under different prediction lengths (3, 6, 9, 12).

shows the comparison of our proposed model and Informer between the predicted values and ground truth when the prediction length is 12. We choose Informer as comparison because it performs better compared with other models (as shown in Table 7.3). We randomly select four variables from the ECG dataset. Each subgraph corresponds to one variable. The x-coordinate represents the prediction time step and the y-coordinate represents the value. As shown in Figure 7.7, the prediction performance of DeepCN is better than that of Informer and it is able to fit the curve of ground truth except for some sudden changes.



(a) Analysis on MAE



(b) Analysis on RMSE

Figure 7.6: Multi-step forecasting error result analysis (MAE and RMSE) of DeepCN and four baseline models on Traffic dataset under different prediction lengths (3, 6, 9, 12).

7.5.5 Analysis

Study of the Coupling Mechanism. This part mainly addresses how the coupling mechanism affects the accuracy and efficiency of our proposed model. We conduct experiments by imposing a different number of couplings orders L (e.g., second-order coupling, third-order coupling, fourth-order coupling, and so on) on the Traffic and ECG datasets, respectively. We separately analyze the relationship between training time and the number of coupling orders, and the relationship between error results (MAE and RMSE) and the number of coupling orders. The result on ECG dataset is shown in Figure 7.8 and the result on Traffic data is shown in Figure 7.9. In Figure 7.8, we analyze the accuracy, the efficiency, and the training loss of our model on ECG dataset under different

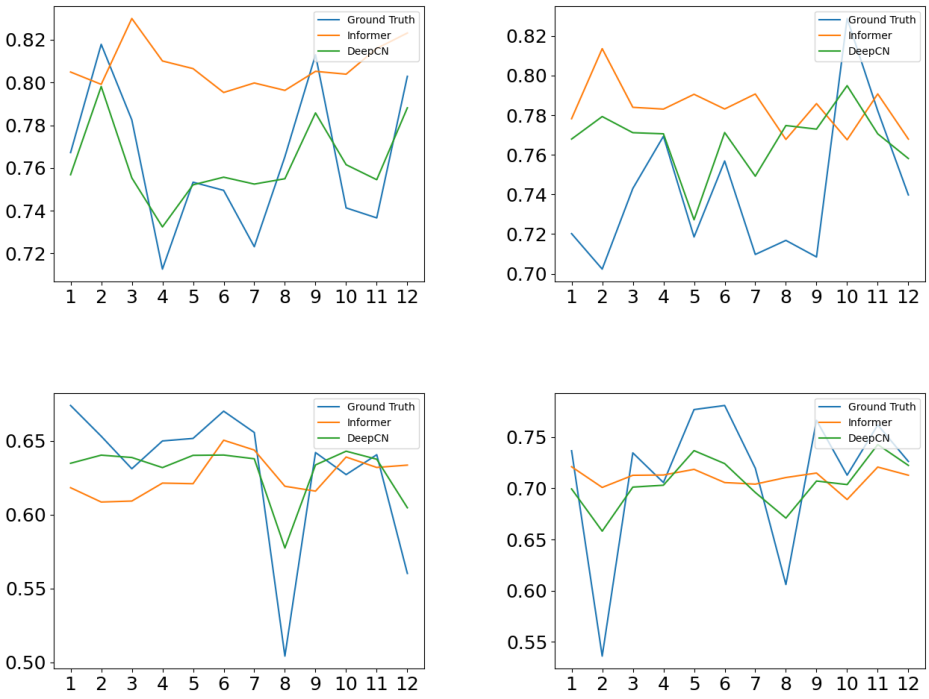


Figure 7.7: The predictions (steps=12) of DeepCN and Informer on ECG dataset.

orders, respectively. (a) The error result (MAE and RMSE) under different orders. (b) The average epoch traing time under different orders. (c) The training loss in different epoch under different orders. Accordingly, we can see that: 1) As the number of orders L increases, the training time becomes longer, but it has little effect on the convergence speed of training. 2) When the order is greater than 2, the accuracy is almost unchanged because the data on ECG dataset has weak correlations among time series. Namely, the data on ECG dataset does not exhibit high-order couplings characteristics. In Figure 7.9, we analyze the accuracy, the efficiency, and the training loss of our model on Traffic dataset under different orders, respectively. (a) The error result (MAE and RMSE) under different orders. (b) The average epoch training time under different orders. (c) The training loss in different epoch under different orders. Accordingly, we can find that: 1) With the number of orders L increasing, the training time becomes longer and the convergence rate of training does not change much. 2) The accuracy rate increases first which shows that a higher-order relationship exists in the traffic dataset. Then with L increasing, the performance worsens since higher order leads to overfitting. From Figure 7.8(a) and Figure 7.9(a), we can conclude that for the strong inter-series correlation time

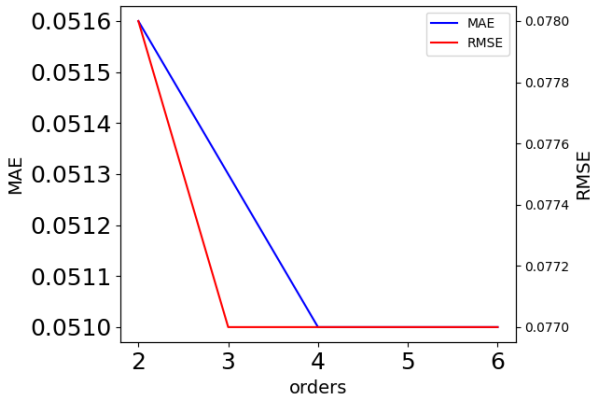
Dataset	Metrics	DeepCN	w/o coupling	w/o 1st	w/o 2st	w/o 3st	w/o 4st
Traffic	MAE	**0.011**	0.013	0.012	0.012	0.012	0.012
	RMSE	**0.022**	0.024	0.024	0.023	0.023	0.022
ECG	MAE	**0.050**	0.051	0.055	0.051	0.051	0.051
	RMSE	**0.076**	0.077	0.081	0.077	0.077	0.077

Table 7.6: Error results (MAE and RMSE) under different orders of couplings

series dataset (e.g., traffic dataset), it is necessary to model the high-order couplings, while for the weak inter-series correlation dataset (e.g., ECG dataset), modeling for the high-order couplings does not bring improvements in effectiveness. This also explains that sequential models achieve generally better results than GNN-based models on ECG dataset, while on Traffic dataset, GNN-based models perform better than other models.

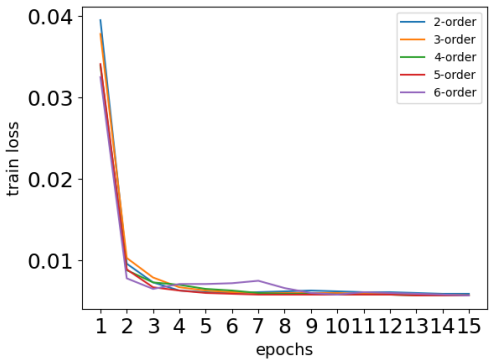
Ablation Study. To further analyze the different order of couplings, we conduct more experiments on the Traffic dataset and ECG dataset to evaluate the effectiveness of different orders. We set the input length and prediction length to 12, and the number of orders L is set to 4. Other experimental settings is the same as introduced in Section 7.5.3. We evaluate the effectiveness of different order of coupling through masking corresponding order. The results are shown in Table 7.6. In the table, **w/o coupling** means that the model is without the coupling mechanism. Namely, the model is only composed of the first order coupling and the fully-connected network. **w/o 1st** represents the coupling mechanism without the first order coupling and we mask the first order coupling from the relationships C. In the similar fashion, **w/o 2st**, **w/o 3st** and **w/o 4st** represents the coupling mechanism without the second order, third order and fourth order couplings, respectively. From the table, we can find that: 1) For traffic dataset, each order of coupling is indispensable. 2) For ECG dataset, compared with other order of couplings, the first order coupling is more important. It shows as well the high-order couplings should be considered in the strong correlation data while the low-order coupling is enough for the relationship modeling in the weak correlation data.

Efficiency Analysis. To evaluate the efficiency of DeepCN, we compare the training time and parameter counts of our proposed model with GNN-based models (StemGNN, AGCRN and MTGNN) and Transformer-based models (Autoformer and Informer) on Wiki and Traffic datasets, respectively. We use the same input length ($T = 12$) and prediction length ($\tau = 12$) for the analysis in the five methods and the results are shown in Table 7.7. From the table, we can find that: 1) the Transformer-based models require



(a) accuracy analysis

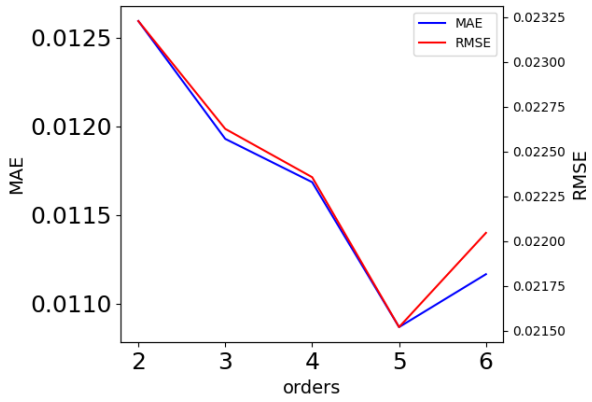
(b) efficiency analysis



(c) train loss analysis

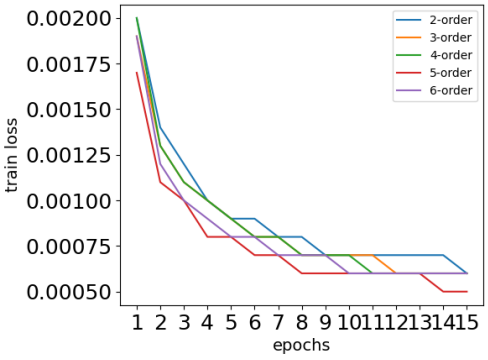
Figure 7.8: Study of coupling mechanism on ECG dataset.

less training time than GNN-based models and our model since they only model the intra-series dependencies while GNN-based models and our model consider both intra-



(a) accuracy analysis

(b) efficiency analysis



(c) train loss analysis

Figure 7.9: Study of coupling mechanism on Traffic dataset.

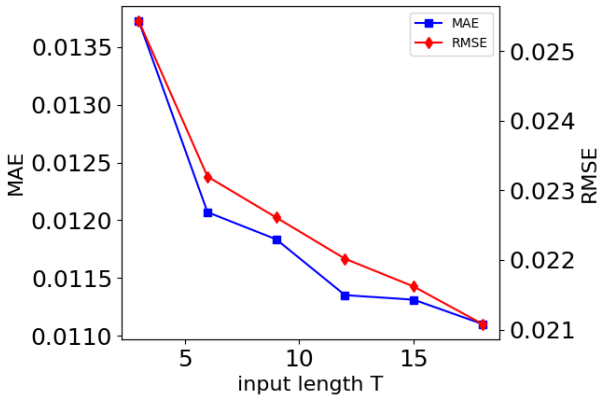
and inter-series dependencies. 2) Compared with the GNN-based models, our proposed model performs more efficiently because the complexity of our model is $\mathcal{O}(N \times T)$ while the

	Training time (s/epoch)					
	DeepCN	StemGNN	AGCRN	MTGNN	Autoformer	Informer
Wiki	9.61	92.59	22.48	27.76	2.39	2.64
Traffic	62.52	201.69	166.61	169.34	14.48	12.99
	Parameters					
	DeepCN	StemGNN	AGCRN	MTGNN	Autoformer	Informer
Wiki	7.91M	4.10M	0.76M	1.53M	15.6M	14.9M
Traffic	8.74M	3.88M	0.75M	1.48M	15.4M	14.8M

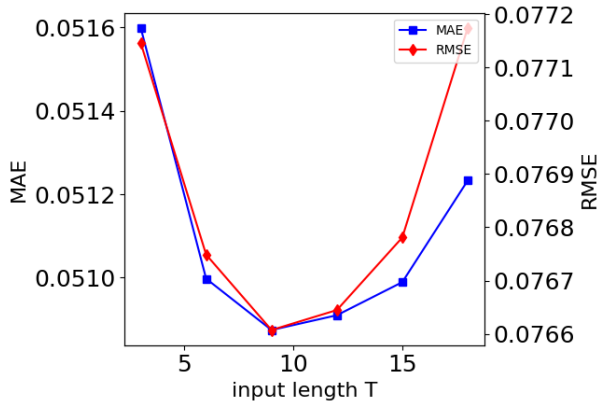
Table 7.7: Results of efficiency analysis on Wiki dataset (variables=1000, samples=803) and Traffic dataset (variables=962, samples=10560).

AGCRN and MTGNN are $\mathcal{O}(N^2)$ and StemGNN is $\mathcal{O}(N^3)$. 3) Transformer-based models consume more parameters than GNN-based models since the self-attention mechanism needs high memory usage. 4) The parameter counts of our model is larger than GNN-based models because our model encode not only timestamp-wise and variable-wise interactions, but also the multi-order couplings. Moreover, our model simultaneously models the intra- and inter-series dependencies while GNN-based models separately model them.

Parameter Sensitivity. We perform parameter sensitivity tests of input length T and embedding size d on Traffic and ECG datasets. All parameters of our model under study are held constant except the input length and the embedding size. (1) *Input length*. The input length reflects the time lag effects and affects the final accuracy. We turn over it with the value $\{3,6,9,12,15,18\}$ for Traffic and ECG datasets, and the result is shown in Figure 7.10 where we compare error results (MAE and RMSE) under different input lengths on Traffic and ECG datasets, respectively.. Figure 7.10(a) shows that with the input length increasing, the accuracy becomes better since the long input length can bring more information and this also demonstrates the relationships between variables have time lag effects which are introduced in Section 7.3. Figure 7.10(b) shows that with the input length increasing, the performance first improves and then decreases due to data redundancy or overfitting. (2) *Embedding size*. The embedding size affects the representation ability and we choose the embedding size over the set $\{128,256,512,768,1024,1280\}$ for Traffic dataset and $\{128,256,512,1024,2048\}$ for ECG dataset. We choose different value set for the two datasets because of the memory limit. In Figure 7.11, we compare error results (MAE and RMSE) under different embedding sizes on Traffic and ECG datasets, respectively. Figure 7.11(a) demonstrates that the



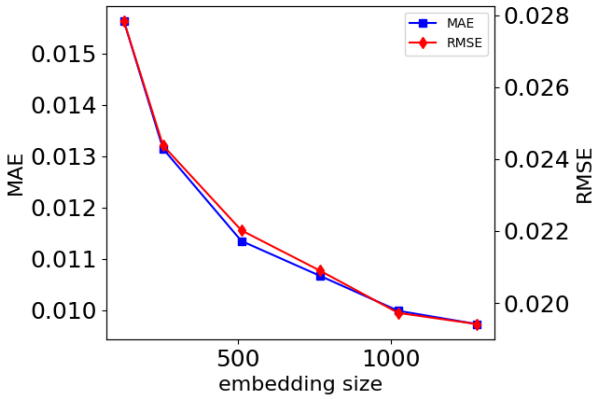
(a) Traffic dataset



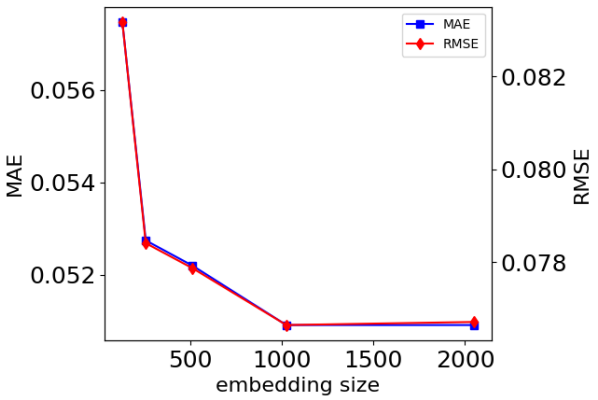
(b) ECG dataset

Figure 7.10: Parameter sensitivity analysis about input length.

performance becomes better with the increase of embedding size while Figure 7.11(b) shows that the performance first improves and then keeps almost unchanged. Unlike the data on ECG dataset, the data on Traffic dataset exhibits strong couplings between the data. Then compared with ECG dataset, the data on Traffic dataset has stronger time lag effects (comparing Figure 7.10(a) with Figure 7.10(b)), and needs bigger embedding size to represent the complicated relationships (comparing Figure 7.11(a) with Figure 7.11(b)).



(a) Traffic dataset



(b) ECG dataset

Figure 7.11: Parameter sensitivity analysis about the embedding size.

7.6 Conclusions

In this chapter, we aim to capture inter-/intra-series couplings to improve the MTS forecasting accuracy. The main contributions are summarized as follows:

- Compared with previous models, we theoretically revisit the relationships among time series, and construct our model based on the couplings which can bring more comprehensive information to enhance the representations of the relationships among time series.

- We design a coupling mechanism to explicitly explore the complicated intra- and inter-series correlations among time series by learning various couplings. The

coupling mechanism explores the diverse and hierarchical couplings between different combinations of time lags of time series with linear complexity.

- We propose a DeepCN model for MTS forecasting which captures complex multivariate relationships based on the coupling mechanism to address the relationships modeling issues on MTS data.
- Extensive experimental results on five real-world datasets show our DeepCN improves an average of 8.9% on MAE and 10.2% on RMSE than the baselines. In addition, more analysis about the coupling mechanism further reveals that why different models perform differently on different datasets which give us enlightenment for handling different types of MTS data.

Part IV

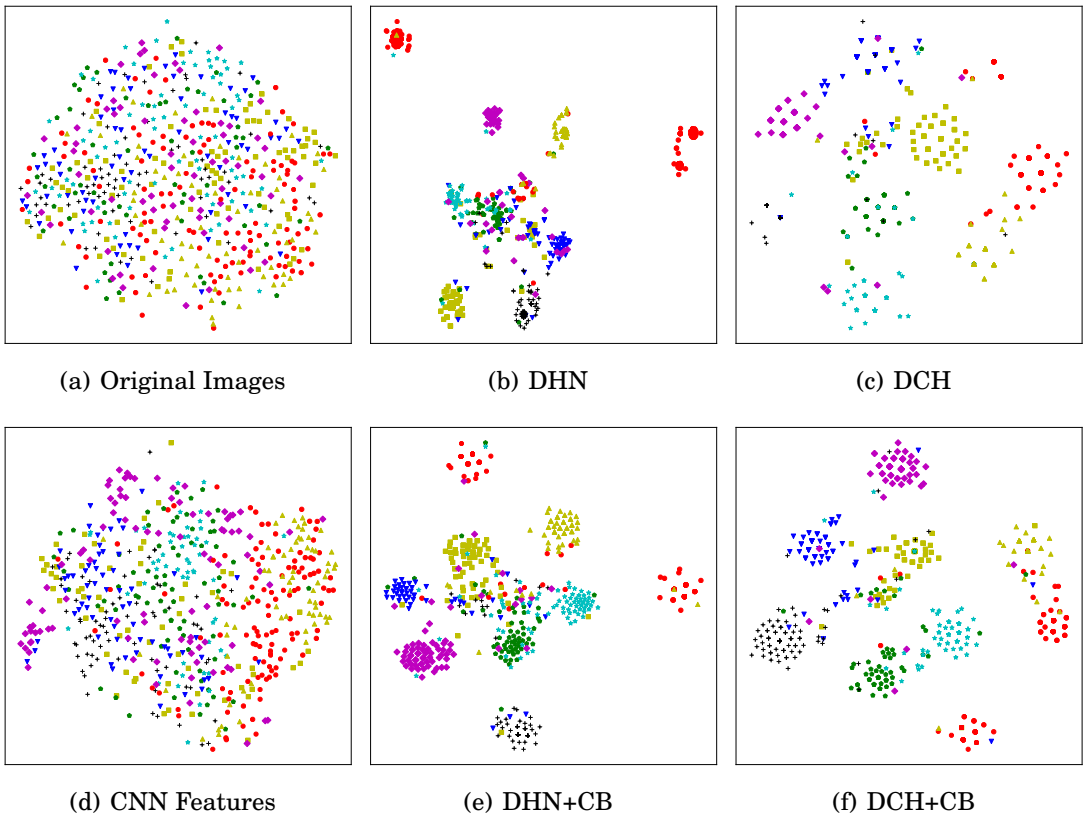
Non-IID Learning to Hash

CHAPTER 8

DEEP SUPERVISED HASHING WITH COMPACTNESS AND INFORMATIVENESS ENHANCEMENT

8.1 Introduction

Benefiting from the advances of deep neural models in nonlinear end-to-end representation learning, deep hashing has enjoyed wide attention in similarity retrieval [234]. Especially, deep supervised hashing (DSH) methods jointly learn deep representation and hash codes through given similarity/label supervision, achieving state-of-the-art retrieval performance [242]. Recent DSH methods focus primarily on refining or customizing objective functions for preserving similarity, e.g., reducing quantization loss caused by continuous relaxation [58], weighting training pairs to tackle the label imbalance issue [26] and introducing class-wise learning objective [228]. However, these methods hardly involve *code balance* constraints to improve hash quality, e.g., compactness and informativeness of hash codes. In fact, a large number of works have demonstrated that code balance can effectively avoid learning collapse, i.e., generating the same hash codes for all similar datapoints, and facilitate generating compact and informative hash codes [33, 238]. Figure 8.1 illustrates an example of random 5,000 images from the CIFAR-10 dataset. Figures (a) and (d) are the visualization of original images and CNN features extracted via a pre-trained AlexNet [108]. Figures (b) and (c) visualize the hash codes generated by DHN [297] and DCH [26] respectively, while figures (e) and (f) visualize the hash codes from DHN and DCH considering code balance constraint



(a) Original Images (b) DHN (c) DCH

(d) CNN Features (e) DHN+CB (f) DCH+CB

Figure 8.1: Visualization illustration using T-SNE.

[222] (suffixed "+CB"). We can observe: figure (b) is overly intensive and figure (c) shows excessive dispersion and intra-cluster overlaps (fewer points in figure), and both two figures show massive inter-cluster overlaps. In contrast, figures (e) and (f) show better performance due to preserving both inter-cluster and intra-cluster discriminability, indicating that code balance promotes code compactness and informativeness.

Unfortunately, traditional code balance constraints, such as bit balance and bit uncorrelation, imposed on whole data population [238], are unsuitable for DSH methods due to three key issues: 1) finding the zero-mean-thresholded hash functions that achieve bit balance is difficult, especially when building deep hash functions [220]; 2) obtaining the hash codes for all datapoints once is both inefficient and impractical when simultaneously learning deep data representation and hash functions; 3) traditional code balance constraints are unsuitable for the conventional batch training in deep models, since guaranteeing code balance across the entire data population hardly hold same balance conditions in each data batch. Due to the above issues, DSH methods rarely consider the benefits obtainable from code balance, which leads to low quality and representability of

hash codes.

To tackle the above issues, we propose a novel probabilistic code balance constraint suitable for DSH scenarios. Specifically, we force each hash code to independently satisfy a discrete uniform distribution on $\{-1,1\}^K$, i.e., $Uni(\{-1,1\}^K)$. Our theoretical analysis indicates that it not only covers the traditional bit balance and bit uncorrelation constraints but maximizes the mutual information between the original data and the corresponding hash codes. The deep insights reveal that the constraint introduces random noise to uniformly scatter datapoints into hash space and to improve hash robustness and avoid overfitting. In addition, we propose a Wasserstein regularization that utilizes the Wasserstein-1 distance to measure the distance between the hash code distribution and the target discrete uniform distribution and minimize the regularization to achieve the constraint.

8.2 Problem Formulation

We first outline the general supervised hashing settings used to achieve similarity-preserving hash codes. Let $\mathscr{X} \subset \mathbb{R}^D$ and $\mathscr{Y} \subset \{-1,1\}^K$ be the input domain and binary hash domain respectively, where d and k denote their respective dimensions. We have pairwise supervision of similarity information $\mathbf{S} \in \{0,1\}^{n \times n}$ for n datapoints where $s_{ij} = 1$ if datapoints $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathscr{X}$ are semantically similar and $s_{ij} = 0$ otherwise. Supervised hashing aims to learn a mapping function $H_\phi := \mathscr{X} \rightarrow \mathscr{Y}$ with parameters ϕ (e.g., a neural network) by minimizing the gap between the similarities $\mathbf{S}$ in the input domain $\mathscr{X}$ and those calculated in the hash domain $\mathscr{Y}$. We then introduce the two traditional code balance constraints.

Code Balance. To avoid severe overfitting and guarantee high-quality hash codes [221, 222], bit balance and bit uncorrelation are usually considered from the information-theoretic perspective. Let $\mathbf{X} := \{\mathbf{x}_i \in \mathscr{X}\}_{i=1}^n$ and $\mathbf{Y} := \{\mathbf{y}_i = H_\Phi(\mathbf{x}_i), \mathbf{y}_i \in \mathscr{Y}\}_{i=1}^n$.

- **Bit Balance:** To generate compacted hash codes, it is desirable to maximize the information contained in each hash bit. According to the maximum entropy principle, hash bits that provide balanced partitioning of $\mathbf{X}$, i.e., $\sum_{i=1}^n \mathbf{y}_i = \mathbf{0}$, have maximum information.

- **Bit Uncorrelation:** A general method of obtaining informative hash codes is to maximize the informativeness in hash codes by forcing the different hash bits to

be uncorrelated (mutually orthogonal), i.e., $\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T = n\mathbf{I}$ (where $\mathbf{I}$ is an identity matrix of size n).

8.3 Probabilistic Code Balance Constraint

We propose a novel code balance constraint that suits DSH scenarios. More specifically, we force each generated hash code independently to satisfy a discrete uniform distribution on $\{-1,1\}^K$, e.g., $\mathbf{y} \sim Uni(\{-1,1\}^K)$, where we can prove that the constraint over $\mathbf{Y}$ covers the aforementioned bit balance and bit uncorrelation constraint:

Theorem 8.1 (Coverage of Bit Balance and Bit Uncorrelation)**.** *For any $\mathbf{y}_i \in \mathbf{Y}$, if $\mathbf{y}_i \sim Uni(\{-1,1\}^K)$, it satisfies $E(\sum_{i=1}^{n} \mathbf{y}_i) = \mathbf{0}$ and $E(\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T) = n\mathbf{I}$.*

Proof. From $\mathbf{y}_i \sim Uni(\{-1,1\}^K)$, we get $\forall y_{ij} \in \mathbf{y}_i, y_{ij} \sim Uni(\{-1,1\})$. Then, we have $E(y_{ij}) = 0$, $E(y_{ij}^2) = 1$ and $E(y_{ik}y_{jk}) = 0$, where $i \neq k$. Accordingly, we get:

$$E(\sum_{i=1}^{n} \mathbf{y}_i) = \sum_{i=1}^{n} E(\mathbf{y}_i) = \mathbf{0}. \tag{8.1}$$

Let $\mathbf{y}_{i\cdot} \in \{-1,1\}^n$ denotes the i-th column of $\mathbf{Y}$. When calculating $\mathbf{Y}^T\mathbf{Y}$, we have diagonal elements:

$$E(\mathbf{y}_{i\cdot}^T \mathbf{y}_{i\cdot}) = \sum_{j=1}^{n} E(y_{ij}^2) = n, \tag{8.2}$$

and for non-diagonal elements,

$$E(\mathbf{y}_{i\cdot}^T \mathbf{y}_{j\cdot}) = \sum_{k=1}^{n} E(y_{ik}y_{jk}) = 0, s.t., i \neq j \tag{8.3}$$

Therefore, $E(\mathbf{Y}^T\mathbf{Y}) = n\mathbf{I}$, i.e., $E(\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T) = n\mathbf{I}$. ■

Theorem 8.1 indicates that $\mathbf{y} \sim Uni(\{-1,1\}^K)$ can achieve both effects of the above two code balance constraints. Unlike the two constraints performing summation over the hash codes of all datapoints in a symmetrical manner, our proposed asymmetrical constraint forces the hash codes to satisfy the independent discrete uniform distribution and has no need to obtain the hash codes for all datapoints beforehand. It can effectively avoid the difficulty in achieving the zero-mean threshold in the bit balance.

Theorem 8.2 (Mutual Information Maximization)**.** *Given $\mathbf{X}$ and $\mathbf{Y}$ where $\mathbf{Y}$ is a deterministic function of $\mathbf{X}$, if $\mathscr{P}_{\mathbf{Y}}$ is a uniform distribution on the space of $\mathbf{Y}$; then mutual information between $\mathbf{X}$ and $\mathbf{Y}$, i.e., $\mathbb{I}(\mathbf{X},\mathbf{Y})$, is the maximum.*

Proof. Since $\mathbf{Y}$ is a deterministic function of $\mathbf{X}$, the conditional entropy $\mathbb{H}(\mathbf{Y}|\mathbf{X})$ is constantly zero. Accordingly, $\mathbb{I}(\mathbf{X},\mathbf{Y}) = \mathbb{H}(Y) - \mathbb{H}(Y|X) = \mathbb{H}(Y)$. Since entropy $\mathbb{H}(\mathbf{Y})$ is always upper bounded by the entropy of the uniform distribution on the space of $\mathbf{Y}$, $\mathbb{I}(\mathbf{X},\mathbf{Y})$ reaches the maximum when $\mathscr{P}(\mathbf{Y})$ confirms a uniform distribution. ■

The key insight underpinning the probabilistic code balance constraint is that of maximizing the mutual information between the original data and the corresponding hash codes. The obtained hash codes, therefore, preserve not only the similarity relationships but also more information from the original data. Intuitively, the constraint uniformly scatters datapoints into hash space via introducing random noises, which facilitates avoiding overfitting training data and improves the generalization robustness of hash functions. In addition, we can easily find that our proposed constraint is irrelevant to the scale of data since the proposed constraint is imposed on every single bit. It is therefore consistent to impose the constraint on data batches and the whole data population, indicating that the constraint is suitable for batch training. In summary, the proposed code balance tackles the three key issues in adopting the traditional code balance constraints and is suitable for DSH. Next, we minimize the distance between the distribution of the generated hash codes and the target discrete uniform distribution to achieve the proposed constraint.

8.3.1 Wasserstein Regularization

We propose a Wasserstein regularization to estimate the distance between the distribution of generated hash codes and the target discrete uniform distribution, thereby achieving the proposed probabilistic code balance constraint via minimizing the estimated distance. We here denote the discrete uniform distribution as $\mathscr{P}_r : Uni(\{-1,1\}^K)$, and the distribution of hash codes as $\mathscr{P}_\phi$, i.e., $\mathbf{y}_i \sim \mathscr{P}_\phi$, which is generated by the specific deep hash function H_ϕ. To achieve $\mathbf{y}_i \sim Uni(\{-1,1\}^K)$, we minimize the distance between the two distributions $\mathscr{P}_r$ and $\mathscr{P}_\phi$. Accordingly, we introduce the Wasserstein-1 distance (a.k.a. Earth-Mover distance), which is a distance function between probability distributions defined on the same metric space Ω, i.e., $\mathscr{P}_r$ and $\mathscr{P}_\phi$ defined on space $\Omega = \{-1,1\}^K$ in our case:

$$W(\mathscr{P}_r,\mathscr{P}_\phi) = \inf_{\gamma\in\Gamma(\mathscr{P}_r,\mathscr{P}_\phi)} \mathbb{E}_{(\mathbf{y},\mathbf{y}')\sim\gamma}\|\mathbf{y}-\mathbf{y}'\|, \tag{8.4}$$

where $\Gamma(\mathscr{P}_r,\mathscr{P}_\phi)$ denotes the set of all joint distributions $\gamma(\mathbf{y},\mathbf{y}')$, the marginal distributions of which are $\mathscr{P}_r$ and $\mathscr{P}_\phi$ respectively. Intuitively, we can understand the definition

via considering the optimal transport problem: in the given space Ω, the Wasserstein-1 distance reflects the minimal cost of transporting mass from $\mathscr{P}_r$ to $\mathscr{P}_\phi$ in order to transform the distribution $\mathscr{P}_r$ to the distribution $\mathscr{P}_\phi$. Analogously, we propose an empirical estimate of the Wasserstein-1 distance in a way that makes it unnecessary to directly estimate the distribution of hash codes. More specifically, given input data $\mathbf{X} \in \mathscr{R}^{n \times D}$, we first randomly sample n binary target vectors denoted by $\mathbf{A}$, the elements of which follow $\mathscr{P}_r$:

$$\mathbf{A} \in \{-1,1\}^{n \times K}, \text{ s.t. } \mathbf{a} \in \{-1,1\}^K, \mathbf{a} \in \mathbf{A}, \mathbf{a} \sim \mathscr{P}_r. \tag{8.5}$$

We then optimally pair the hash code of each datapoint with each target vector respectively such that the sum of the distances between all pairs is minimal. To obtain the optimal pairing matrix $\mathbf{P} \in \{0,1\}^{n \times n}$, we first define a set of constraints (denoted $\mathbb{P}$) for all possible pairing matrices:

$$\mathbb{P}_n = \{\mathbf{P} \in \{0,1\}^{n \times n} | \mathbf{P}\mathbf{1}_n = \mathbf{1}_n, \mathbf{P}^T\mathbf{1}_n = \mathbf{1}_n\}, \tag{8.6}$$

where $\mathbf{1}_n$ denotes a n-sized vector with all 1s. Given hash codes $\mathbf{Y}$ of $\mathbf{X}$, we then optimize the following objective:

$$\min_{\mathbf{P} \in \mathbb{P}_n} \frac{1}{2} ||\mathbf{Y} - \mathbf{P}\mathbf{A}||_F^2 = \min_{\mathbf{P} \in \mathbb{P}_n} -\mathbf{tr}(\mathbf{P}\mathbf{A}\mathbf{Y}^T), \tag{8.7}$$

where we use a squared ℓ_2 distance, and $\mathbf{tr}(\cdot)$ denotes the trace function. Once an optimal pairing matrix $\mathbf{P}$ is found, the above distance can be regarded as an estimate of the Wasserstein-1 distance between $\mathscr{P}_r$ and $\mathscr{P}_\phi$, that is:

$$\inf_{\gamma \in \Gamma(\mathscr{P}_r, \mathscr{P}_\phi)} \mathbb{E}_{(\mathbf{y}, \mathbf{y}') \sim \gamma} \|\mathbf{y} - \mathbf{y}'\| \approx \min_{\mathbf{P} \in \mathbb{P}_n} -\mathbf{tr}(\mathbf{P}\mathbf{A}\mathbf{Y}^T). \tag{8.8}$$

Accordingly, the learning objective of DSH equipped with Wasserstein Regularization (WR) contains two components: the *similarity loss* used to preserve the similarity $\mathbf{S}$ in the original space, and the *Wasserstein Regularization* which enhances the compactness and informativeness of hash codes:

$$\mathscr{J}(\phi) = \min_{\phi} \ell(\mathbf{S}, \mathbf{Y}\mathbf{Y}^T) + \beta \min_{\phi} \min_{\mathbf{P} \in \mathbb{P}_n} -\mathbf{tr}(\mathbf{P}\mathbf{A}\mathbf{Y}^T), \tag{8.9}$$

where $\mathbf{Y} = H_\phi(\mathbf{X})$, and $\beta > 0$ denotes a balance weight adjusting the importance of *Wasserstein Regularization*. The similarity loss is calculated via the loss function ℓ and can be specified to the similarity loss of a certain DSH method.

Algorithm 3 Alternating Optimization

Input: Given input data $\mathbf{S}$.
Initiate the neural network ϕ and set batch size b
while *stopping criteria is not satisfied* **do**
  Fixing ϕ, randomly sample b input samples $\mathbf{X}_b$ and calculate $\tilde{\mathbf{Y}}_b$.
  Randomly sample $\mathbf{A}_b$ from the distribution $\mathscr{P}_r$.
  Solve $\mathbf{P}_b$ using Hungarian algorithm.
  Update ϕ using batch gradient descent according to the gradients $\nabla_\phi \mathscr{J}'(\phi)$.
end while

8.3.2 Optimization

Following the common continuous relaxation treatment [26, 117], we approximate the sgn function with a squashing function (e.g., tanh), the output $\tilde{\mathbf{Y}}$ of which is within $(-1,1)$. We thus obtain a differentiable neural function $H'_\phi := \mathbf{X} \rightarrow \tilde{\mathbf{Y}}$ and the corresponding learning objective $\mathscr{J}'(\phi)$ updated as below, of which the parameters can be solved using gradient-based back propagation algorithm.

$$\mathscr{J}'(\phi) = \min_{\phi} \ell(\mathbf{S}, \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^T) - \beta \min_{\phi} \min_{\mathbf{P}\in\mathbb{P}_n} \mathbf{tr}(\mathbf{P}\mathbf{A}\tilde{\mathbf{Y}}^T). \tag{8.10}$$

Apart from $\mathbf{A}$ which is randomly sampled from the distribution $\mathscr{P}_r$, we further need to obtain the optimal pairing matrix $\mathbf{P}$. Obtaining $\mathbf{P}$ is a linear assignment problem, which can be solved exactly via Hungarian algorithm [109]. Under the batch gradient updates, the algorithm can efficiently perform under the restriction of one batch, significantly reducing the time complexity from $O(n^3)$ to $O(nb^2)$, where b is the number of samples in one batch.

Accordingly, we apply an alternating optimization to solve Equation 8.10. First, when fixing $\tilde{\mathbf{Y}}$, we sample b training samples $\mathbf{X}_b$ and calculate its corresponding intermediate matrix $\tilde{\mathbf{Y}}_b$. We then randomly sample b target vectors, denoted by $\mathbf{A}_b$ from the discrete uniform distribution and solve $\mathbf{P}_b \in \mathbb{P}_b$ with the Hungarian algorithm to optimally pair $\tilde{\mathbf{Y}}_b$ and $\mathbf{A}_b$. Finally, when fixing $\mathbf{A}_b$ and $\mathbf{P}_b$, we update the parameters ϕ via batch gradient descent. The corresponding algorithm is presented in Algorithm 3.

8.4 Experiments and Evaluation

To verify the effectiveness of the proposed probabilistic code balance in promoting retrieval performance and improving hash code quality, we select six state-of-the-art

deep hashing baselines and compare the baselines with their variants equipped with WR on two public image datasets.

8.4.1 Experimental Setup

Datasets. We adopt two public image datasets for evaluation. 1) **CIFAR-10**[1]: the dataset consists of 60,000 32×32 images in 10 classes, where each class has 6,000 images. Two images will be treated as a ground-truth similar pair if they share common label. 2) **NUS-WIDE**[2]: it contains a total of 269,648 images. Similar to [130, 297], we use its subset of 195,834 images associated with the 21 most frequently concepts, where each concept consists of at least 5000 images, and define two images as a groundtruth similar pair if they share at least one common label.

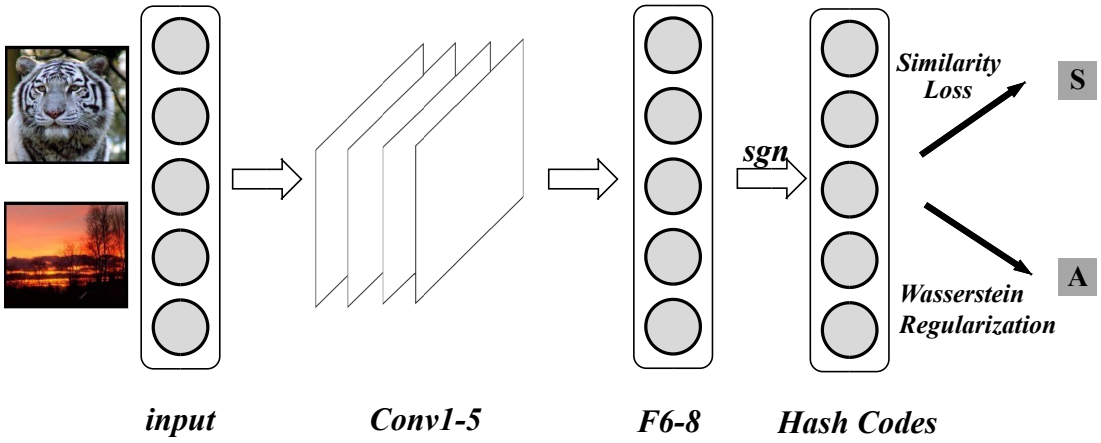


Figure 8.2: Neural network architecture used in the experiments.

8.4.2 Network Structure

To facilitate fair comparison, we adopt the CNN architecture, AlexNet [108], for all baselines and their WR-enabled variants. Specifically following the architecture of AlexNet, we utilize five convolutional layers (*Conv1-5*) and three fully connected layers *F6-8*, and apply signature function, i.e., sgn, to generate the binary hash codes based the output of the layer *F8*, as shown in Figure 8.2. We then utilize the hash codes to calculate the similarity loss (precisely the loss used in the baselines) and Wasserstein regularization.

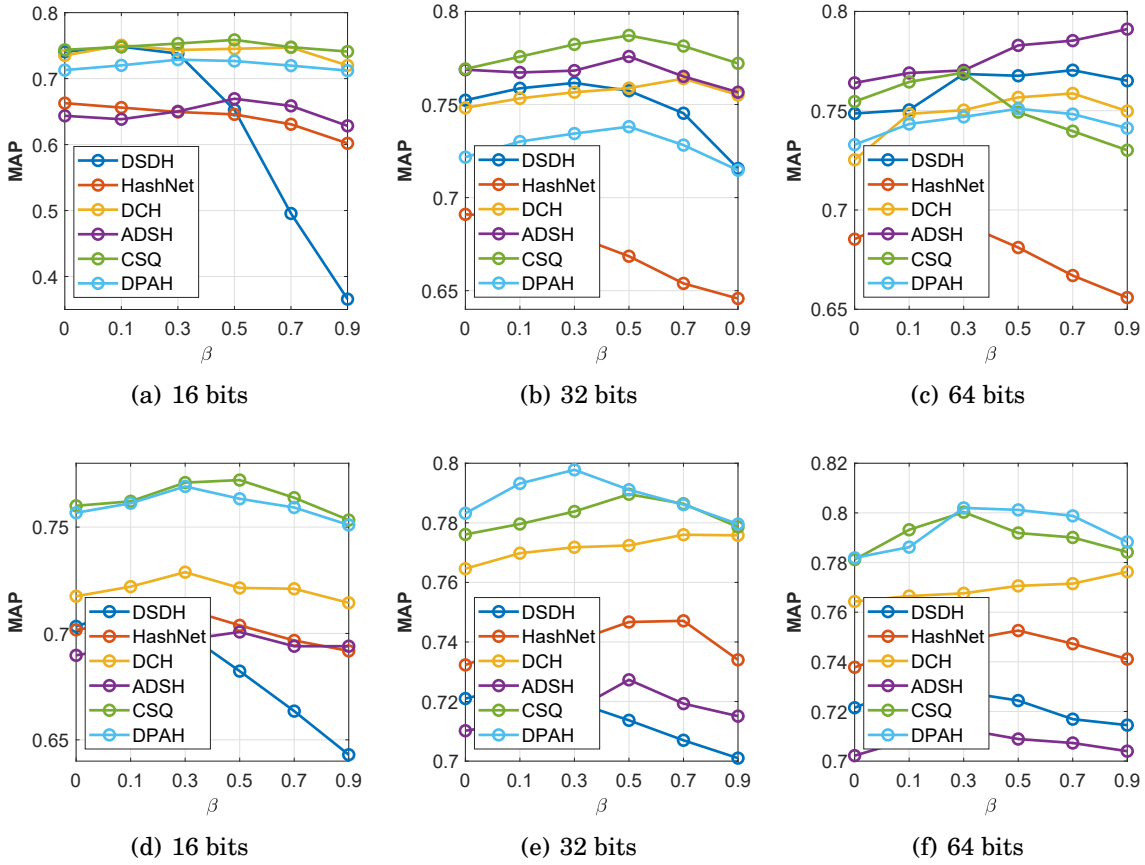
[1] http://www.cs.toronto.edu/kriz/cifar.html
[2] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

In the training phase, we initialize the network with pretrained parameters from ImageNet and apply the continuous relaxation treatment and approximate the sgn function with a squashing function (e.g., tanh).

Table 8.1: MAP evaluation of the six baselines and their WR-enabled variants on two public datasets.

The better results between baselines with (+) and without (/) WR are shown in bold where $*$ indicates the statistically significant improvement (i.e., two-sided t-test with $p < 0.05$). Δ denotes the average improvement of each WR-enabled variant over its baseline.

Method	WR	CIFAR-10					NUS-WIDE				
		16 bits	32 bits	48 bits	64 bits	Δ	16 bits	32 bits	48 bits	64 bits	Δ
DSDH	/	0.7407	0.7523	0.7478	0.7486	2.52	0.7038	0.721	0.7208	0.7215	1.37
	+	**0.7493***	**0.7737***	**0.7704***	**0.7713***		**0.7127***	**0.7315***	**0.7301***	**0.7321***	
HashNet	/	0.6628	0.691	0.6903	0.6853	2.29	0.7017	0.7323	0.7418	0.7378	1.75
	+	**0.6771***	**0.7045***	**0.7083***	**0.7021***		**0.7115***	**0.7467***	**0.7539***	**0.7526***	
DCH	/	0.7451	0.7484	0.7491	0.7253	1.77	0.7182	0.7555	0.7593	0.7413	3.08
	+	**0.7515***	**0.7626***	**0.7587***	**0.7477***		**0.7291***	**0.7718***	**0.7887***	**0.7763***	
ADSH	/	0.6438	0.7612	0.764	0.761	3.67	0.7007	0.7102	0.7182	0.7022	2.61
	+	**0.6702***	**0.7869***	**0.7911***	**0.7893***		**0.7144***	**0.7273***	**0.7358***	**0.7278***	
CSQ	/	0.7436	0.7691	-	0.755	2.05	0.76	0.7761	-	0.7812	1.93
	+	**0.7587***	**0.7871***	-	**0.7685***		**0.7721***	**0.7896***	-	**0.8003***	
DPAH	/	0.7129	0.7217	0.7347	0.7329	2.33	0.7567	0.7832	0.7912	0.7819	2.11
	+	**0.7287***	**0.7381***	**0.7521***	**0.751***		**0.7691***	**0.7978***	**0.8097***	**0.802***	



(a) 16 bits (b) 32 bits (c) 64 bits

(d) 16 bits (e) 32 bits (f) 64 bits

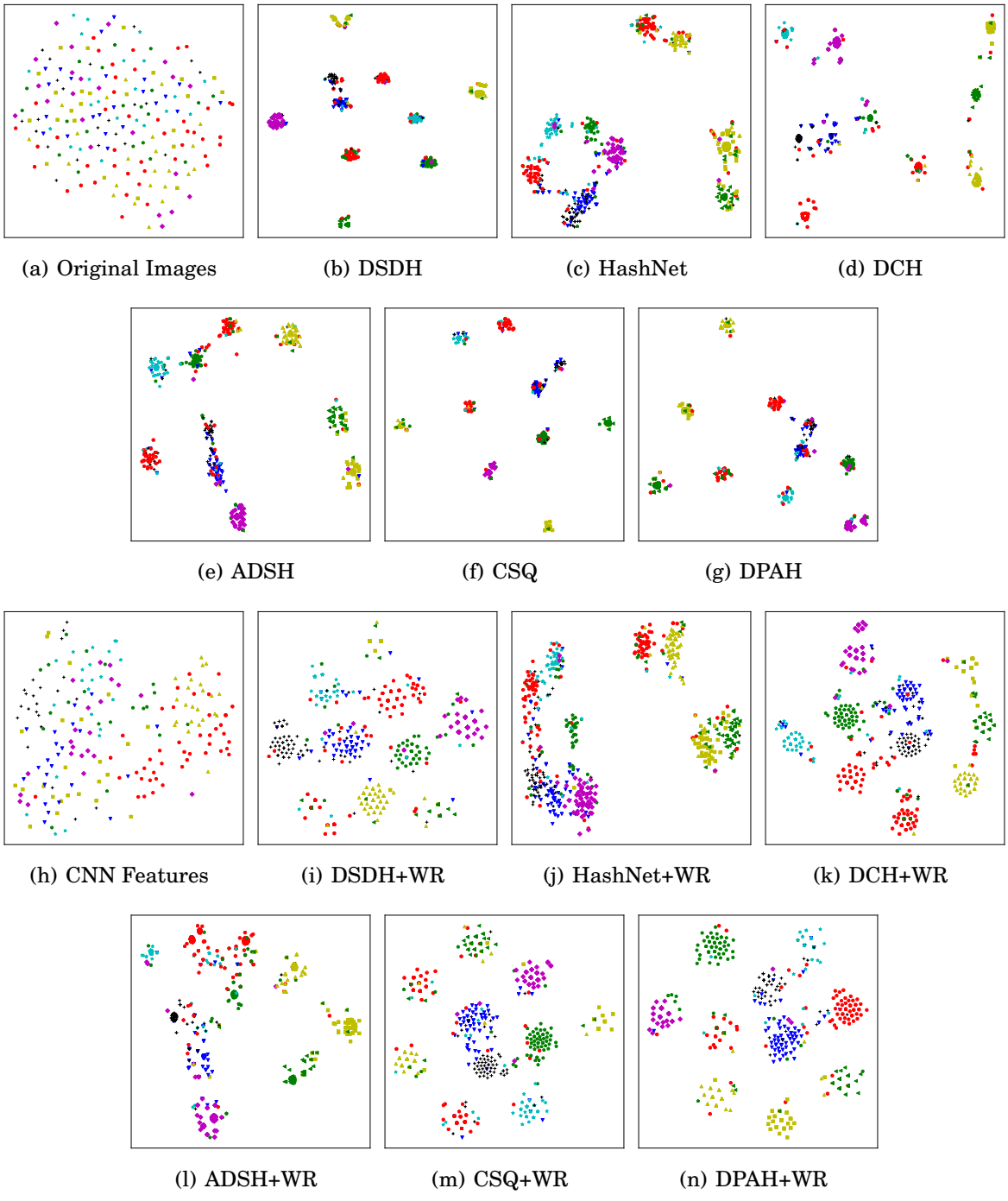
Figure 8.3: MAP evaluation of the WR-enabled variants with different β on CIFAR-10 (a-c) and NUS-WIDE (d-f).

Baselines. We select different types of SotA DSH methods, including pointwise, pairwise and class-wise methods, for evaluation in our experiments: 1) DSDH [117] jointly learns a linear classifier based on pointwise groundtruth labels along with the hash functions; 2) HashNet [27] tackle the data balance issue by weighting training pairs; 3) DCH [26] further introduces a Cauchy cross-entropy loss to measure pairwise similarity; 4) ADSH [93] directly learns hash codes for all database points asymmetrically and efficiently. 5) CSQ [268], the latest class-wise method, proposes a global similarity metric referring to hash centers. 6) DPAH [228] introduces learnable class centers as the global proxies to capture global similarity. Note that non-deep hashing methods are not included in our experiments due to the focus on DSH. To evaluate the effectiveness of our proposed probabilistic code constraint, we construct a WR-enabled variant for each baseline, i.e., adding the Wasserstein regularization to its objective function.

Evaluation Protocol. We follow the experimental settings recommended in [117, 297]. In CIFAR-10 and NUS-WIDE, we randomly sample 100 images per class to form the testing set, with the remaining images used as the database, then randomly sample 100 images and 500 images per class from the database to act as the validation set and training set respectively (we adopt the whole database for training in ADSH for consistency.). To facilitate fair comparison, we adopt the same network as shown in Table 8.2, fix batch size to 256 for all baselines, and evaluate the comparative methods over code length $K \in \{16, 32, 48, 64\}$. We tune the baselines on validation sets to find their optimal configuration. For their WR-enabled variants, we further tune the balance hyperparameter β within the range of 0.1 to 1.0 with step 0.1 to obtain the best results. All experiments have been run five times, and the average results are reported. In the experiments, we report the Mean Average Precision (MAP) to evaluate similarity retrieval performance, while Mutual Information Neural Estimation (MINE) [10] is used to evaluate the informativeness of hash codes.

8.4.3 Results and Discussion

Retrieval Performance. The MAP results of each baseline and its corresponding WR-enabled variant are reported in Table 8.1. As we can observe from the table, the WR-enabled variants outperform their corresponding baselines. More specifically, the WR-enabled variants perform much better on longer code lengths, i.e., 32 bits and 48 bits, where the MAP improvement of each WR-enabled variant over its baseline is over 1.74% and can reach 3.87%. This is intuitively attributable to the fact that the learning of the similarity-preserving objective is more easily distorted, and more vulnerable to the noise introduced by WR, as smaller code lengths. Moreover, we find that the improvement on ADSH, CSQ, and DPAH is larger than that on the others. This is attributable that WR-enabled ADSH trained on the database points improves the quality of the hash codes on these points, while the Wasserstein regularization can further improve the discriminability of the hash codes (especially for similar datapoints) generated from the class-wise methods, i.e, CSQ and DPAH. The above results are impressive since the Wasserstein regularization promotes the retrieval performance on different kinds of DSH methods, including point-wise (label-based), pairwise and class-based methods. In addition, we investigate the approximation quality of Wasserstein-1 distance (i.e., Equation 8.8) and observe that the WR-enabled variants get better performance with the increase of batch size and the best MAP when batch size $b = 512$ (see *Appendix D*). These results demonstrate that Wasserstein regularization introducing random noises is



(a) Original Images (b) DSDH (c) HashNet (d) DCH

(e) ADSH (f) CSQ (g) DPAH

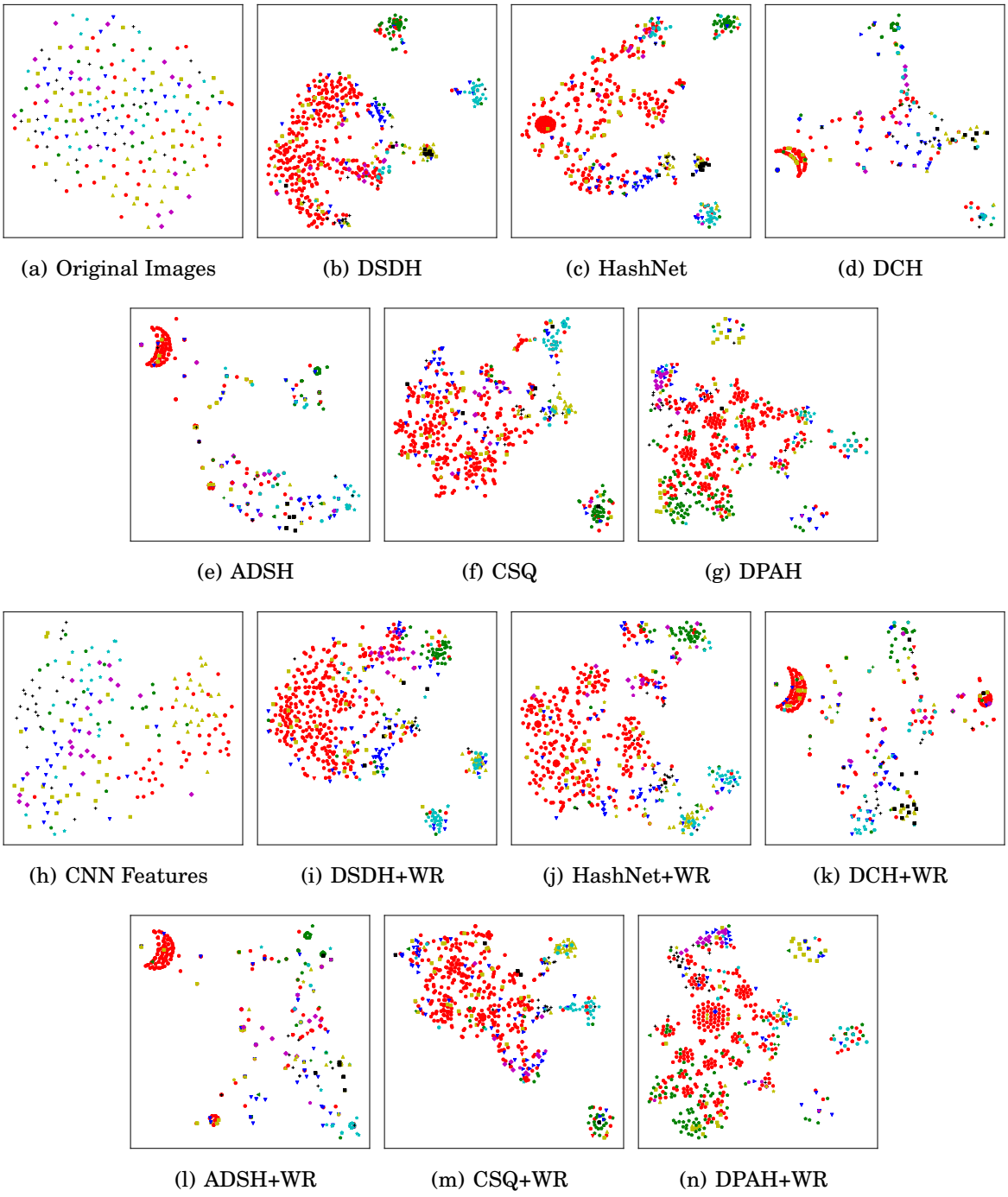
(h) CNN Features (i) DSDH+WR (j) HashNet+WR (k) DCH+WR

(l) ADSH+WR (m) CSQ+WR (n) DPAH+WR

Figure 8.4: Visualization of the hash codes of the testing set on CIFAR-10.

We report one pair visualization for each baseline, where each pair corresponds to a baseline and its WR-enabled variant with the same code length.

beneficial to improving the robustness of hash functions and avoiding overfitting training data.



(a) Original Images (b) DSDH (c) HashNet (d) DCH

(e) ADSH (f) CSQ (g) DPAH

(h) CNN Features (i) DSDH+WR (j) HashNet+WR (k) DCH+WR

(l) ADSH+WR (m) CSQ+WR (n) DPAH+WR

Figure 8.5: Visualization of the hash codes of the testing set on NUS-WIDE.

Training Complexity. We ran the baselines and their WR-enabled variants using a single GTX-1080 GPU. The average training time costs on different training batch sizes are reported in Table 8.2; Δ denotes the percentage of additional training time cost

WR	16	32	64	128	256	512
/	5.9	4.14	3.23	2.77	2.57	2.47
+	6.1	4.29	3.43	2.95	2.89	2.96
Δ	3.39	3.62	6.19	6.5	12.45	19.84

Table 8.2: Average training time cost (seconds per epoch) over six baselines with $K = 48$ on CIFAR in terms of different batch sizes.

required by WR-enabled variants over their corresponding baselines. The results show that the time cost for computing Wasserstein regularization increases as the batch size increases. This is because the bulk of the time cost associated with WR computation is linked with predicting hash codes for batch data, i.e., $O(bT_h)$ where T_h denotes the time complexity for a single datapoint, as well as that for applying the Hungarian algorithm, i.e., $O(nb^2)$. In addition, the percentage is less than 13% when the batch size $b <= 256$, indicating that the additional time cost is affordable and worthwhile considering the benefits of WR.

Performance Varying with β. We also investigate the retrieval performance of WR-enabled variants under varying values of β on the two datasets. Note that results on $K = 48$ are not reported due to space limitations. Comparing the figures of different bits, we find that the WR-enabled variants perform better on longer code lengths and worse on shorter code lengths, confirming the results in Table 8.1. In addition, the performance of most WR-enabled variants decreases as the values of β increase from 0.3 to 0.9, especially on $K = 16$ and $K = 32$. This is reasonable because Wasserstein regularization – i.e., the probabilistic code balance – improves hash quality by introducing random noise and avoiding DSH overfitting from arising during the supervision of similarity in training data. However, when β increases, a larger weight is allocated to Wasserstein regularization; this may result in the hash function being highly biased to noises and greatly degrade the retrieval performance.

Performance Varying with Batch Size The MAP results of the WR-enabled variants with code length $K = 36$ over different batch sizes are shown in Table 8.3. The variants get better performance with the increase of batch size and achieve the best MAP when batch size $b = 512$. Comparing the results in Section *Retrieval Performance* and those in Table 8.3, we find that the WR-enabled baselines outperform the baselines when $b >= 128$ and perform bad when $b <= 64$. Since we apply the Hungarian algorithm in

the batch training to empirically estimate the Wasserstein-1 distance on training data, a larger batch size results in a more accurate estimate and is beneficial to learn hash codes conforming to the probabilistic code balance constraint. The above results confirm that the code balance effectively improves the performance of similarity retrieval. Note that as the batch size increases, the similarity retrieval performance improves, and training cost increases as well. Therefore, the balance between retrieval performance and training efficiency should be considered when selecting the batch size.

Table 8.3: MAP evaluation of WR-enabled six baselines with $K = 32$ on CIFAR. The best results on each method are shown in bold.

method	Enabled by WR				
	32	64	128	256	512
DSDH+	0.757	0.764	0.77	0.774	**0.781**
HashNet+	0.688	0.695	0.698	0.705	**0.71**
DCH+	0.754	0.759	0.762	0.763	**0.766**
ADSH+	0.763	0.77	0.778	0.787	**0.795**
CSQ+	0.769	0.772	0.779	0.787	**0.791**
DPAH+	0.713	0.717	0.721	0.738	**0.747**

8.4.4 Code Compactness and Informativeness

In this section, we evaluate the compactness and informativeness of the hash codes generated in the first experiment. We first calculate the mutual information between the original data (images) and the corresponding hash codes via MINE by a $100-100-1$ sized three-layer fully-connected neural network, as in Table 8.4. The results show that WR enables large amounts of mutual information between each original data and its hash codes, which indicates that the Wasserstein regularization improves the informativeness of the generated hash codes. We further visualize the hash codes via T-SNE, as shown in Figure 8.4 and Figure 8.5. We find that WR-enabled variants achieve good performance, and even outperform their baselines, in terms of similarity preservation. In addition, when comparing WR-enabled variants with their baselines, we easily observe that hash codes generated by WR-enabled variants are scattered and exhibit less overlap, indicating the higher compactness and informativeness of the hash codes. These results demonstrate the effectiveness of the probabilistic code balance and reveal that WR, with its introduction of random noise to scatter hash codes, improves the compactness

Method	WR	CIFAR-10		NUS-WIDE	
		16 bits	48 bits	16 bits	48 bits
DSDH	/	0.0652	0.2421	0.3988	0.4351
	+	**0.2589**	**0.8625**	**0.4855**	**0.5774**
HashNet	/	0.3265	-0.1598	0.4032	0.3284
	+	**0.4158**	**0.8124**	**0.6889**	**0.4302**
DCH	/	-0.4109	-1.3194	0.2854	0.1212
	+	**2.3706**	**-0.8522**	**0.6512**	**0.3622**
ADSH	/	-0.9106	-0.1058	0.3681	-0.0159
	+	**0.463**	**0.1225**	**0.7014**	**0.0249**
CSQ	/	-0.2578	-0.2235	0.1481	0.2545
	+	**0.273**	**0.3131**	**0.687**	**0.5318**
DPAH	/	-0.8566	-0.5169	0.2171	0.0711
	+	**0.1669**	**0.643**	**0.9378**	**0.321**

Table 8.4: Informativeness evaluation of the baselines and their WR-enabled variants. Better results are marked in bold.

and informativeness of hash codes and provides benefits for model generalization, i.e., avoiding training data overfitting.

8.5 Conclusions

To improve the quality of hash codes in deep supervised hashing, we propose a novel probabilistic code balance constraint, which forces each hash bit to independently satisfy a discrete uniform distribution on $\{-1, 1\}^K$ (K is hash code length). We prove and analyze the effectiveness and insights of the proposed constraint, and further incorporate the Wasserstein regularization to implement the constraint. Extensive experiments by comparing six DSH baselines and their WR-enabled variants on two image benchmark datasets demonstrate the probabilistic code balance can effectively promote retrieval performance and improve the compactness and informativeness of hash codes.

In this chapter, we consider the coupling of original inputs and hash codes to enhance the quality of hash codes, specifically code compactness and informativeness. The contributions can be summarized as follows:

- We propose a novel probabilistic code balance constraint suitable for DSH and

theoretically analyze the effectiveness and insights of the constraint for the first time.

- We introduce Wasserstein regularization to achieve the constraint via Wasserstein-1 distance and propose an empirical estimate of the Wasserstein regularization.
- We conduct extensive experiments using different types of SotA DSH baselines on two benchmark datasets in terms of metrics on retrieval performance and informativeness. The results show the constraint not only enhances code compactness for promoting retrieval performance but improves the informativeness of hash codes.

CHAPTER 9

DEEP SUPERVISED HASHING FOR HIGH-DIMENSIONAL AND HETEROGENEOUS CASE-BASED RETRIEVAL AND CLASSIFICATION

9.1 Introduction

Case-based reasoning (CBR) is an incremental learning methodology of analogy solution making, inspired by cognitive science that humans handle new problems by referring to past analogous experiences (cases) [277]. Generally, a complete CBR system includes four key steps: case retrieval, case reuse, case revision, and case retention. Case retrieval assesses the similarity between a target case and past cases and obtains the most similar past cases. Case reuse suggests a solution for the target case according to the solutions of past cases. Case revision verifies the correctness of the suggested solution and revises the solution if necessary. Case retention maintains the solved cases into a case base for future problem-solving. Due to its good interpretability and practicability, CBR has been applied successfully to classification [277], diagnosis [9], decision support [64], fault detection [256], and various fields, e.g., finance [39, 184], industry [28, 100, 114], manufacturing [99, 123], and medicine [112, 152].

Similarity measures play a decisive role in obtaining similar cases and thus largely affect the performance of CBR systems. Appropriate similarity measures should maximize the model fitness to data characteristics and explore the inherent data characteristics for handling the underlying problems [140]. However, enormous complex data, specifically

high-dimensional and heterogeneous data, has penetrated every corner of our lives, bringing significant challenges to quantifying data characteristics and building accurate similarity measures. In addition, the increasing amount of cases retained during the incremental learning of CBR often leads to a huge knowledge base (case base) with enormous cases and bring a huge computation burden of similarity calculation and ranking in case retrieval. The aforementioned issues usually degrade the performance and efficiency of CBR systems. Intuitively, it is crucial to probe the intrinsic characteristics of high-dimensional and heterogeneous data and build efficient data-aware similarity measures, where approaches such as approximate nearest neighbor search (e.g., hashing techniques studied in this work) are promising to improve the retrieval efficiency.

Measuring the similarity of large amounts of high-dimensional data is critical for data mining and machine learning tasks and applications, e.g., information retrieval [287], clustering [42], and hashing [191]. The volume and complexity of such data usually entails prohibitively large storage and time consumption. Handcrafting an accurate similarity measure is challenging since it is usually the case that only partial unknown features are relevant to the task at hand [131]. Therefore, many traditional similarity measures, e.g., Euclidean distance, cosine similarity, and Jaccard coefficient, perform poorly or even work out of action under high dimensionality. To address these issues, existing research usually projects high-dimensional data into a low-dimensional space by dimensionality reduction like manifold learning [52] or principal component analysis [301] and then learns an approximate similarity measure in the reduced space. However, high-dimensional data is often accompanied by the issue of sparsity where many useful features are rarely observed, and datapoints are scattered in multiple lower dimensional manifolds. It brings new challenges that these dimensionality reduction techniques may be inapplicable and parameters for the reduction easily lead to severe overfitting to the data.

In addition to the high-dimensionality issue, it is also crucial to consider the heterogeneity embodied in complex cases in similarity assessment. In heterogeneous data, attributes may follow different distributions and show different significance. Intuitively, this may lead to the inaccuracy of most handcrafted similarity measures, which adopt consistent attribute similarity or linear (e.g., average) aggregation functions [154]. To tackle the heterogeneity of case data in the real world, recent studies usually utilize different similarity measures on different types of attributes, e.g., using the Euclidean distance for numerical attributes and the Hamming distance for categorical attributes [173], and then optimize the allocation of attribute weights and aggregation functions for similar-

ity assessment [2, 277, 278]. Those methods optimize similarity measures by linear or non-linear aggregation functions in a data-independent manner, which cannot depict the complex attribute coupling relationships and heterogeneity among attributes [139]. Accordingly, a more promising but challenging approach is to capture the heterogeneity of attributes [41, 81, 285] and learn data-aware similarity metrics [156, 217] to capture feature couplings [38, 161, 293].

The CBR problem-solving process mainly includes the successive execution of case retrieval and case reuse since case revision and retention are usually performed offline. Case retrieval is the most time-consuming step which usually consists of traversing all cases for similarity calculation and ranking the cases according to their similarity scores. With the increasing number of cases maintained in the case base, the retrieval efficiency and storage burden of CBR become increasingly critical and affect the applicability of CBR systems. To improve the efficiency, recent studies focus on reducing the search space by building indices to structurally organize the case base [128, 182] or partition and index cases by clustering [277] and mapping [69, 295]. However, these methods usually rely on domain expertise and similarity measures to partition and index the case base and need additional time costs to update the case base and maintain its indexing structure. This may greatly degrade the performance and efficiency of CBR systems on high-dimensional and heterogeneous data where domain expertise can hardly penetrate.

Another promising way to accelerate similarity retrieval is approximate nearest neighbor search such as locality-sensitive hashing (LSH) [87, 281]. LSH maps similar datapoints (cases) into same 'buckets' (encoded with low-dimensional binary codes) with high probability, which preserves the similarity relationships between datapoints and can be regarded as a way of dimensionality reduction on high-dimensional data [44]. Several studies introduce LSH into CBR systems to approximate the nearest neighbor search process and scale traditional CBR systems to large-scale data [88, 89, 239]. The studies show CBR systems equipped with hashing techniques can greatly improve retrieval efficiency and achieve desirable performance with expected loss in accuracy. Recently, a tremendous amount of research shows that data-dependent hashing (a.k.a., learn to hash) methods perform significantly better than data-independent hashing (e.g., LSH) [190, 222, 298]. Data-dependent hashing learns similarity-preserving compact hash codes from data with/without (supervised/unsupervised) given similarity information by various flexible hash functions, e.g., PCA [220], kernel functions [190], and deep neural networks [242]. Naturally, data-dependent hashing facilitates to capture complex data characteristics and potentially improves the performance of hashing-based CBR systems,

which, however, has not yet been introduced into CBR.

To address the above challenges and gaps, we introduce supervised deep hashing in CBR systems and propose a hash function with an adaptive hashing network to build a Hashing-enabled CBR system (short for HeCBR). Specifically, each feature is represented by the multiplication of its 'amplitude' (position embedding for the feature) and 'frequency' (the feature value). Subsequently, a multilinear interaction layer is introduced to aggregate the feature embeddings to capture multiview feature couplings and obtain case embeddings. The above two embedding layers filter out zero-valued features and calculate the case embeddings in multilinear time complexity to efficiently handle the high-dimensionality and heterogeneity issues. Then, the case embeddings are fed into fully-connected layers and a hash layer to generate the binary hash codes. To learn the hashing network, we construct the similarity groundtruth from the solutions of cases (precisely two cases being similar if they have same/similar solutions, e.g., classification label, dissimilar otherwise). We then optimize the learning objectives of minimizing the loss between the similarity groundtruth and case similarity in the hash space and constrain the loss with a quantization regularizer. Considering the nature of incremental learning in CBR, we further propose a mechanism of incremental learning combining an adaptive learning objective and an update strategy to update the hash function and hash codes respectively for retaining solved cases. Accordingly, we construct a hashing-enabled CBR model to integrate the adaptive hashing network to index the case base and accelerate the similarity retrieval speed. The overview problem-solving process of HeCBR is shown in Figure 9.1.

9.2 Problem Formulate

Supervised learning has been prevalent and successful in achieving high-quality semantic hash codes. Below, we outline the general settings in supervised hashing. Assume $\mathscr{X} \subset \mathbb{R}^d$ and $\mathscr{Y} \subset \{-1,1\}^r$ refer to the input (original) space and binary hash space respectively, where d and r denote their respective dimensions. Let us denote the given/-calculated pairwise supervision of similarity information $\mathbf{S} \in \{0,1\}^{n \times n}$ for n data points where $s_{ij} = 1$ if data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathscr{X}$ are semantically similar, and $s_{ij} = 0$ otherwise. The aim of supervised hashing is to learn a mapping function $H_\phi := \mathscr{X} \rightarrow \mathscr{Y}$ with parameters ϕ (e.g., a neural network) by minimizing the gaps between the similarity in the input space and that in the hash space.

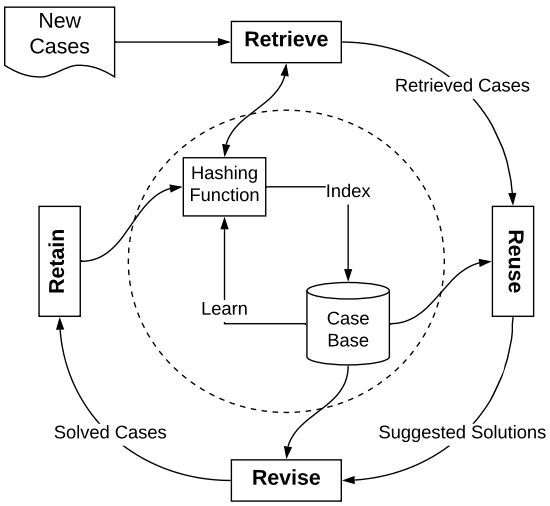


Figure 9.1: The problem-solving process of case-based reasoning with adaptive hashing.

9.3 Deep Hashing Network

Taking advantage of recent advances in deep hashing, we introduce a deep hashing network tailored for high-dimensional and heterogeneous data. The network architecture is shown in Figure 9.2 which contains four components: feature embedding, multiview feature interaction, fully-connected layers, and a hashing layer. Next, we introduce the components in detail.

9.3.1 Feature Embedding

Due to the high dimensionality and heterogeneity of case data, it is challenging to directly feed the feature vector of each case into neural networks. Inspired by the idea of performing feature embedding in [275], we introduce position embedding to represent heterogeneous features in a unified distributional space to tackle the above issues. Specifically, assume $\mathbf{X} := \{\mathbf{x}_i \in \mathscr{X}\}_{i=1}^n$ as the input data of n cases, and let $\mathbf{x}_i = [x_{i1}, x_{i2} \cdots, x_{id}]^T \in \mathbb{R}^{d \times 1}$ denote the feature vector of the i-th case, i.e., $\mathbf{x}_i \in \mathbf{X}$. Note that all categorical features in the raw feature vector are converted to binary features by one-hot encoding. Given the j-th feature in the i-th case, i.e., $x_{ij} \in \mathbf{x}_i$ where $i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, d\}$, we have:

$$\mathbf{e}_{ij} = \begin{cases} \mathbf{0}, & x_{ij} = 0 \\ x_{ij}\mathbf{w}_j^p, & x_{ij} \neq 0 \end{cases} \tag{9.1}$$

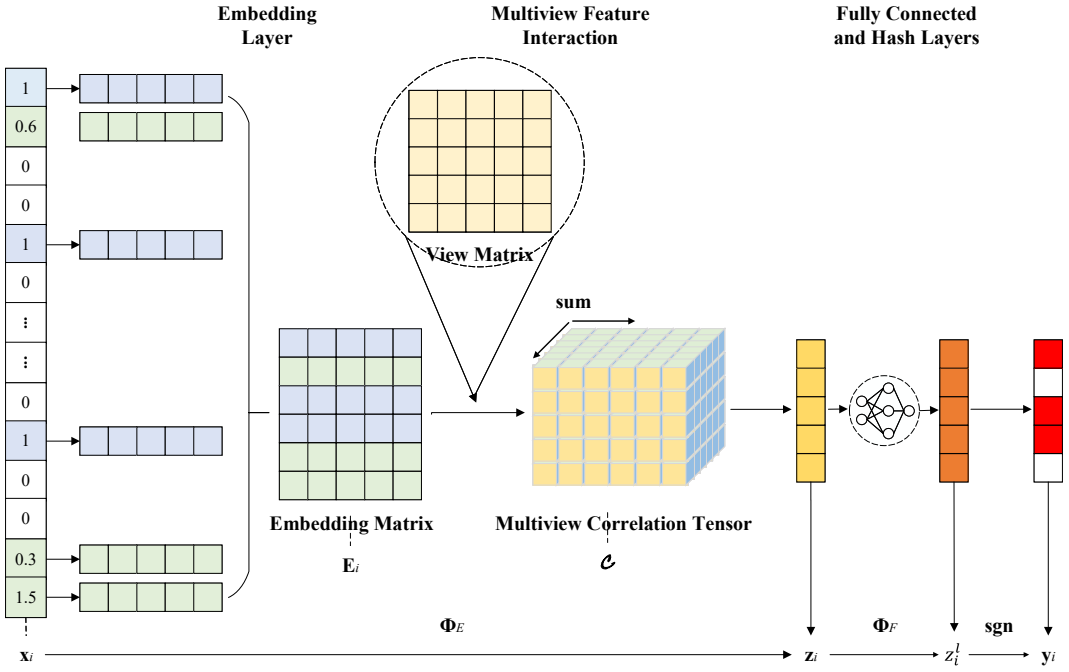


Figure 9.2: The architecture of deep hashing network.

where $\mathbf{e}_{ij}$ denotes the embedding vector for the feature x_{ij}, and $\mathbf{w}_j^p \in \mathbb{R}^{k_w \times 1}$ denote the embedding vector for the j-th position and k_w denotes the embedding dimension. Here, we introduce a position embedding matrix $\mathbf{W}^p = \{\mathbf{w}_1^p, \mathbf{w}_2^p, \cdots, \mathbf{w}_d^p\} \in \mathbb{R}^{k_w \times d}$ to represent each feature position, where $\mathbf{w}_j$ corresponds to the j-th position in each feature vector. Accordingly, we have $\mathbf{E}_i = \{\mathbf{e}_{i1}, \cdots, \mathbf{e}_{id}\} \in \mathbb{R}^{k_w \times d}$ as the feature embedding matrix for case $\mathbf{x}_i$.

According to Equation (9.1), the embedding vector of each feature is the multiplication between its feature value and position embedding vector. Intuitively, the position embedding projects heterogeneous features onto the unified space $\mathbb{R}^{k_w}$ where it captures the heterogeneous feature couplings. In addition, since the embeddings of zero-valued (binary or numerical) features are constantly $\mathbf{0} \in \mathbb{R}^{k_w \times 1}$, the resultant large proportion of all-zero vectors in each feature embedding matrix $\mathbf{E}$ facilitates to filtrate zero-valued features and to alleviate the sparsity in high-dimensional cases (see more analysis in Section 9.3.2).

9.3.2 Multiview Feature Interaction

To obtain the representation for each case, we propose a multiview feature interaction module to aggregate feature embeddings. Specifically, we introduce a view matrix $\mathbf{V} \in$

$\mathbb{R}^{k_w \times k_v}$ (k_v to denote the view dimension) and then resort to CANDECOMP/PARAFAC (CP) factorization to calculate the case representation below. Given case x_i, we have:

$$\mathbf{z}_i = \sum_{p=1}^{d} \sum_{q=p+1}^{d} \left[\sum_{j=1}^{k_w} \mathbf{E}_{i,j\cdot} \circ \mathbf{E}_{i,j\cdot} \circ \mathbf{V}_{j\cdot} \right] \Bigg|_{pq\cdot} \tag{9.2}$$

where $\mathbf{E}_{i,j\cdot} \in \mathbb{R}^{d\times 1}$ denotes the j-th row in the feature embedding matrix $\mathbf{E}_i$, $\mathbf{V}_{j\cdot} \in \mathbb{R}^{k_v \times 1}$ is the j-th row in the view matrix $\mathbf{V}$ and $\circ$ denotes the outer product. In Equation (9.2), we calculate the multiview correlations (a tensor) among all features in $\mathbf{x}_i$, i.e., the CP term in the brackets denoted as $\mathscr{C} \in \mathbb{R}^{d\times d\times k_v}$, and we then sum over the correlation matrix to each view[1], i.e., the first two dimensions of the tensor $\mathscr{C}$, to generate the representation vector $\mathbf{z}_i \in \mathbb{R}^{k_v \times 1}$ for case $\mathbf{x}_i$. For convenience, we denote the projection process of feature embedding and multiview feature interaction as $\Phi_E : \mathbf{X} \mapsto \mathbf{Z}$, where $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\} \in \mathbb{R}^{k_v \times n}$.

Let us expand Equation (9.2) and probe the computation of each element in the resultant $\mathbf{z}_i$ for the case $\mathbf{x}_i$:

$$z_{ik} = \sum_{p=1}^{d} \sum_{q=p+1}^{d} \left[\sum_{j=1}^{k_w} v_{jk} \mathbf{E}_{i,j\cdot} \circ \mathbf{E}_{i,j\cdot} \right] \Bigg|_{pq} = \sum_{p=1}^{d} \sum_{q=p+1}^{d} \langle \mathbf{E}_{i,\cdot p}, \mathbf{E}_{i,\cdot q} * \mathbf{V}_{\cdot k} \rangle = \sum_{p=1}^{d} \sum_{q=p+1}^{d} \langle \mathbf{e}_{ip}, \mathbf{e}_{iq} * \mathbf{V}_{\cdot k} \rangle. \tag{9.3}$$

where $\mathbf{E}_{i,\cdot p} \in \mathbb{R}^{k_w \times 1}$ and $\mathbf{V}_{\cdot k} \in \mathbb{R}^{k_w \times 1}$ are the p-th column (i.e., the embedding vector of the p-th feature $\mathbf{e}_{ip}$) and k-th column $\mathbf{V}$ respectively, and $\langle \cdot, \cdot \rangle$ denotes the inner product and $*$ denotes the element-wise product.

From Equation (9.3), we easily find that all-zero vectors (i.e., the embeddings of zero-valued features) are absorbed in the summation and do not count in calculating the embedding vector $\mathbf{z}_i$ for the case $\mathbf{x}_i$. This treatment efficiently extracts informative features from high-dimensional sparse cases. In addition, the multiview feature interaction module calculates the pairwise (second-order) interaction between any two features by inner product and introduces a view matrix to diversify the calculation of feature interactions. Intuitively, the module captures inter-feature couplings between (heterogeneous) features from multiple views referring to the learned view matrix $\mathbf{V}$, accordingly learning diversified feature couplings for high-level layers. In addition, the interaction module can capture first-order interactions by manipulating the feature vector of each case $\mathbf{x}$, specifically concatenating a constant value of 1 with $\mathbf{x}$ and obtaining an extended feature vector $[1, \mathbf{x}]$. In summary, *Feature Multiview Interaction* module has three advantages:

[1]Since the tensor $\mathscr{C}$ is symmetric, the summation is performed upon the upper-right elements of the correlation matrix of each view.

- It learns feature interactions from multiple views to capture diverse inter-feature couplings between heterogeneous features, which is more effective than feed-forward neural networks in capturing intrinsic feature correlations [12, 275].

- It effectively filtrates zero-valued features and generates informative representation for each case, shielding the influence of the high dimensionality and unpredictable sparsity.

- It does not introduce extra parameters except for the view matrix and efficiently performs the calculation with time complexity of $O(dk_w k_v)$. We provide detailed analysis in Section 9.4.5.

9.3.3 Fully-connected and Hash Layers

After obtaining the dense representation vectors for all cases, we feed case representation into full-connected layers and subsequently a hash layer, which learn high-level semantic representations and generate binary hash codes respectively. Specifically, we feed the representation $\mathbf{z}_i$ of any case $\mathbf{x}_i$ into l fully connected layers. Each layer learns a nonlinear mapping:

$$\mathbf{z}_i^{\ell} = \sigma^{\ell}(\mathbf{W}^{\ell}\mathbf{z}_i^{\ell-1} + \mathbf{b}^{\ell}) \tag{9.4}$$

where $\mathbf{W}^{\ell}$ and $\mathbf{b}^{\ell}$ are the weight and bias parameters of the ℓ-th layer, σ^{ℓ} is the corresponding activation function, and $\mathbf{z}_i^{\ell}$ denotes the ℓ-th layer hidden representation of case $\mathbf{x}_i$ (note that $\mathbf{z}_i^0 = \mathbf{z}_i$). For convenience, let us denote the l fully connected layers as $\Phi_F : \mathbf{Z} \mapsto \mathbf{Z}^l$, where $\mathbf{Z}^l = \{\mathbf{z}_1^l, \mathbf{z}_2^l, \cdots, \mathbf{z}_n^l\} \in \mathbb{R}^{r\times n}$ and r denotes the dimension of hash codes. We then obtain the hash code by feeding l-layer output $\mathbf{z}_i^l$ into the hash layer which contains a sign function. Formally, we have the hash code for the case $\mathbf{x}_i$: $\mathbf{y}_i = \mathrm{sgn}(\mathbf{z}_i^l)$ and $\mathbf{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_n\} \in \{-1, 1\}^{r\times n}$. Accordingly, the hash mapping function, i.e., $H_{\phi} : \mathbf{X} \to \mathbf{Y}$, is defined below:

$$H_{\phi}(\mathbf{X}) = \mathrm{sgn}\left(\Phi_F(\Phi_E(\mathbf{X}))\right) = \mathbf{Y} \tag{9.5}$$

where H_{ϕ} is the hashing network shown in Figure 9.2, $\phi = \{\mathbf{V}, \mathbf{W}^p, \mathbf{W}^1, \cdots, \mathbf{W}^l, \mathbf{b}^1, \cdots, \mathbf{b}^l\}$ denotes the network parameters and $\mathrm{sgn}(\cdot)$ denotes the elementwise sign function, which results in -1 if its input is negative, otherwise 1.

9.3.4 Learning Objectives

Generally, the Hamming distance $d_H(\cdot,\cdot)$ is utilized to measure the distance between hash codes, which can be achieved by inner product $\langle\cdot,\cdot\rangle$, i.e., $d_H(\cdot,\cdot)=\frac{1}{2}(r-\langle\cdot,\cdot\rangle)$. Accordingly, we apply the inner product to define the following likelihood function: given a pair of cases $\mathbf{x}_i,\mathbf{x}_j$ and their hash codes $\mathbf{y}_i,\mathbf{y}_j$, to estimate the similarity s_{ij} of them, we have

$$P(s_{ij}|\hat{s}_{ij})=\begin{cases}\sigma(\alpha\hat{s}_{ij}), & s_{ij}=1\\ 1-\sigma(\alpha\hat{s}_{ij}), & s_{ij}=0\end{cases} \tag{9.6}$$

where $\hat{s}_{ij}=\langle\mathbf{y}_i,\mathbf{y}_j\rangle$ denotes the estimated similarity between hash codes, σ is a Sigmoid function to scale the inner product into a distribution, and $\alpha\in(0,1]$ is a scaling hyperparameter to control the bandwidth of Sigmoid function. Smaller α gives rise to a smaller saturation zone, where the Sigmoid function has zero gradient. In addition, the larger $\hat{s}_{ij}$ is, the larger $P(s_{ij}=1|\hat{s}_{ij})$ will be, i.e., a larger similarity between hash codes $\mathbf{y}_i$ and $\mathbf{y}_j$ implies a higher probability of two cases $\mathbf{x}_i$ and $\mathbf{x}_j$ being similar. Accordingly, to achieve the objective of preserving the similarity between cases in the hash space, we maximize the likelihood of all pairs of cases (in the training set):

$$\prod_{s_{ij}\in\mathbf{S}}P(s_{ij}|\hat{s}_{ij})=\prod_{s_{ij}\in\mathbf{S}}\sigma(\alpha\hat{s}_{ij})^{s_{ij}}(1-\alpha\hat{s}_{ij})^{1-s_{ij}} \tag{9.7}$$

Taking the negative logarithm of the likelihood, we have the objective function w.r.t. the cross entropy loss:

$$\begin{aligned}\min_{H_\phi}\mathscr{L}(\mathbf{Y},\mathbf{S})&=\min_{H}-\sum_{s_{ij}\in\mathbf{S}}\log P(s_{ij}|\hat{s}_{ij})\\&=\min_{H_\phi}-\sum_{s_{ij}\in\mathbf{S}}(\alpha s_{ij}\hat{s}_{ij}-\log(1+e^{\alpha\hat{s}_{ij}}))\\&=\min_{H_\phi}-\alpha\mathbf{S}*(\mathbf{Y}^T\mathbf{Y})+\log(1+e^{\alpha\mathbf{Y}^T\mathbf{Y}})\end{aligned} \tag{9.8}$$

where $\hat{s}_{ij}=\langle\mathbf{y}_i,\mathbf{y}_j\rangle=\langle H_\phi(\mathbf{x}_i),H_\phi(\mathbf{x}_j)\rangle$.

Due to the binary constraints in Equation (9.5), we relax the hash code to the final output of the fully-connected layers, i.e., $\mathbf{y}_i\approx\mathbf{z}_i^l$, to approximate the non-differentiable sgn function. Therefore, we have the approximate mapping function: $\bar{H}_\phi(\mathbf{X})=\mathbf{Z}^l=\Phi_F(\Phi_E(\mathbf{X}))$. To squash the l layer representation $\mathbf{Z}^l$ within $[-1,1]$, we encourage $\mathbf{Z}^l$ to be binary by utilizing the Sigmoid-like function: $\sigma^l(x)=2/(1+e^x)-1$. However, the continuous relaxation will cause two important issues: 1) introducing uncontrollable quantization error when binarizing $\bar{H}_\phi(\mathbf{X})$ to $\mathbf{Y}$, and 2) raising approximation error by performing

inner product on $\bar{H}_\phi(\mathbf{X})$ as the surrogate of $\mathbf{Y}$ [297]. To control the quantization error and approximation error, we introduce a quantization regularizer to minimize the difference between the hash codes and their relaxation surrogate shown below:

$$\mathscr{R} = \|\mathbf{Z}^l - \mathbf{Y}\|_2 = \|\bar{H}_\phi(\mathbf{X}) - H_\phi(\mathbf{X})\|_2. \tag{9.9}$$

Theorem 9.1 (Regularizer Upper Bound)**.** *The quantization regularizer on H_ϕ is upper bounded by $nd - \mathbf{tr}(\bar{H}_\phi(\mathbf{X})^T \bar{H}_\phi(\mathbf{X}))$, i.e.,*

$$\|\bar{H}_\phi(\mathbf{X}) - H_\phi(\mathbf{X})\|_2 \leq nd - \mathbf{tr}(\bar{H}_\phi(\mathbf{X})^T \bar{H}_\phi(\mathbf{X})). \tag{9.10}$$

Proof. Since we have $H_\phi(\mathbf{X}) = \mathrm{sgn}(\bar{H}_\phi(\mathbf{X}))$, $H(\mathbf{X})$ and $\bar{H}_\phi(\mathbf{X})$ have the same sign, yielding that

$$\begin{aligned} \|\bar{H}_\phi(\mathbf{X}) - H_\phi(\mathbf{X})\|_2 &= \||\bar{H}_\phi(\mathbf{X})| - |H_\phi(\mathbf{X})|\|_2 \\ &= \sum_{i=1}^{n} \||\bar{H}_\phi(\mathbf{X}_i)| - \mathbf{1}\|_2 \\ &\leq \sum_{i=1}^{n} \|\bar{H}_\phi(\mathbf{X}_i)^T \bar{H}_\phi(\mathbf{X}_i) - d\|_1 \\ &= nd - \mathbf{tr}(\bar{H}_\phi(\mathbf{X})^T \bar{H}_\phi(\mathbf{X})) \end{aligned} \tag{9.11}$$

where **tr** is the trace function. Proved. ■

Accordingly, we approximate hash codes $\mathbf{Y}$ with $\bar{H}_\phi(\mathbf{X})$ in Equation (9.8) and combine the loss function with the upper bound of the quantization regularizer, achieving the final learning objective for the proposed hashing network below:

$$\min_{\bar{H}_\phi} -\alpha \mathbf{S} * \Psi_{\bar{H}_\phi} + \log(1 + e^{\alpha \Psi_{\bar{H}_\phi}}) - \lambda \mathbf{tr}(\Psi_{\bar{H}_\phi}) \tag{9.12}$$

where $\Psi_{\bar{H}_\phi} = \bar{H}_\phi(\mathbf{X})^T \bar{H}_\phi(\mathbf{X})$, $\lambda \in (0,1)$ is a hyperparameter to balance the weights of similarity loss $\mathscr{L}$ and the quantization regularizer $\mathscr{R}$. By minimizing the learning objective in Equation (9.12), we can learn the model parameters of the proposed hashing network, i.e., $\bar{H} = \{\mathbf{W}^p, \mathbf{W}^1, \cdots, \mathbf{W}^l, \mathbf{b}^1, \cdots, \mathbf{b}^l\}$. Besides, hyperparameters $\{k_w, k_v, l, \alpha, \lambda\}$ are selected per empirical experiments and grid search. After the hashing network is well trained, we next introduce the network into case-based reasoning for effective storage and efficient case retrieval. In the stage of case retention, the network is adaptively updated during retaining new data (cases). For a better understanding of the CBR process, we place the detailed introduction to the adaptive update mechanisms in Section 9.4.4.

9.4 Hashing-enabled Case-based Reasoning

Leveraging the proposed hashing network, we propose a Hashing-enabled Case-Based Reasoning (HeCBR) model, in which the hash network is introduced to transform high-dimensional and heterogeneous cases into low-rank binary hash codes. The hash codes are similarity-preserving compact representations for cases, which provides effective and efficient similarity retrieval. As shown in Figure 9.1, next, we introduce each phase of the problem-solving process of HeCBR in detail.

9.4.1 Case Representation

A good case representation not only provides organization and indexing of cases for efficient case retrieval but also facilitates accurate similarity measurement. To achieve this, we adopt our proposed hash network to transform cases into compact hash codes. Based on hash codes, we construct a hash table (a form of an inverted index) to organize and index all cases. The hash table consists of buckets with each bucket indexed by a hash code. For example, if the dimension of hash code r is 8, then there are up to 2^8 buckets with each bucket indexed by an 8 bits hash code. Each case $\mathbf{x}_i$ is then placed into a bucket $H_\phi(\mathbf{x}_i)$. Due to the learned hash codes being similarity-preserving, the hash approach, different from the conventional hashing algorithm avoiding mapping two samples into the same bucket, essentially aims to maximize the probability of collision between similar cases and meanwhile minimize the probability of collision between dissimilar cases. Before constructing the hash table, we need to train the proposed hashing network H_ϕ based on the cases in the case base, where the input feature vectors of cases are given or extracted from the descriptions of the cases and the required similarity relations $\mathbf{S}$ are provided or calculated based on the solutions (labels) of the cases.

9.4.2 Case Retrieval

When a new case comes, we need to retrieve the most similar cases to the new case and leverage the solutions of the retrieved cases to solve the new case. Specifically, to search for the most similar cases to a new case $\mathbf{x}_c$, we first generate the hash code $\mathbf{y}_c$ for the case, i.e., $\mathbf{y}_c = H(\mathbf{x}_c)$. Then we retrieve the cases lying in the bucket indexed by hash code $\mathbf{y}_c$ and treat the cases as the candidates of the most similar cases to $\mathbf{x}_c$. Usually, this is followed by a reranking step: reranking the retrieved candidates according to the true

distances computed using the original features of cases and attaining top-N most similar cases. Note that if no cases exist in bucket $\mathbf{y}_c$, we can retrieve the cases in the nearest buckets (for example those within 2-hamming distance) with bucket $\mathbf{y}_c$ instead, and the indices of the nearest buckets can be obtained by modifying each bit in the hash code $\mathbf{b}_c$ into its alternative value in turn. Since case representation and case retention can be performed offline, case retrieval becomes the most time-consuming phase of CBR and is important to the applicability of CBR methods. Due to the benefits of hash table lookup, similar case candidates can be attained with time complexity of $O(1)$. The remaining time-consuming steps are to generate the hash code for the new case and rerank the candidate cases, which can also be completed in a short time period. We discuss the time complexity in detail in Section 9.4.5.

9.4.3 Case Reuse and Case Revision

During the phase of case reuse, we utilize the retrieved top-N most similar cases to suggest solutions for new cases. A general approach is to design a voting function and suggest the solution with the most votes for each new case. For convenience, we adopt a simple majority voting function where each case denotes one vote and thus the solution supported by most cases will be the optimal one. After suggesting solutions for new cases, we check the actual solutions for the new cases and revise the suggestion if the suggested solutions are different from the actual solutions. Next, the solved cases will be retained in the case base for future problem-solving.

9.4.4 Case Retention

In the phase of case retention, all solved cases will be retained and used to update the hash function (network) and the corresponding hash codes. During the update of the hash function H_{phi}, we need to consider whether or not to update the hash function at all. Accordingly, we decide how to correct the hash function. Inspired by [16], we adopt the hinge loss function to define an offset function:

$$\mathscr{O} = \begin{cases} \max(0, r\beta - \hat{s}_{ij}), & s_{ij} = 1 \\ \max(0, r\beta + \hat{s}_{ij}), & s_{ij} = 0 \end{cases} \tag{9.13}$$

where $\beta \in [0,1]$ is a hyperparameter designating the extent to which the hash function may produce a loss. From Equation (9.13), we can see the larger β is, the larger the loss l_{ij} is, indicating that more rigid similarity is preserved and more information is retained

from the training pair. Imposing the offset function on the similarity loss function, we then obtain the learning objective l_{ij} for adaptive update as follows:

$$l_{ij} = \begin{cases} \max(0, r\beta - \hat{s}_{ij}) \log(1 + e^{-\alpha \hat{s}_{ij}}), \ s_{ij} = 1 \\ \max(0, r\beta + \hat{s}_{ij}) \log(1 + e^{\alpha \hat{s}_{ij}}), \ s_{ij} = 0 \end{cases} \tag{9.14}$$

where we do not consider the quantization regularizer for simplicity since higher β acts as an alternative to the quantization regularizer. When $l_{ij} = 0$, we do not perform any update, which effectively cuts invalid update that has small gradients. In addition, since updating the hash codes (table) is usually time-consuming, we perform the update every $n_u = 100$ new cases to avoid frequently update the hash function and hash codes. The treatment also avoids the update overfitting to certain new cases. The update process can be done offline to improve efficiency.

9.4.5 Complexity Analysis

9.4.5.1 Time Complexity of Multiview Feature Interactions

Recalling Equation (9.3), we reformulate the calculate for each element in $\mathbf{z}_i$ for i-th case. For convenience, we omit the subscript i in the following.

$$\begin{aligned} z_k &= \sum_{p=1}^{d} \sum_{q=p+1}^{d} \langle \mathbf{e}_p, \mathbf{e}_q * \mathbf{V}_{\cdot k} \rangle \\ &= \frac{1}{2} \sum_{p=1}^{d} \sum_{q=1}^{d} \langle \mathbf{e}_p, \mathbf{e}_q * \mathbf{V}_{\cdot k} \rangle - \frac{1}{2} \sum_{p=1}^{d} \langle \mathbf{e}_p, \mathbf{e}_p * \mathbf{V}_{\cdot k} \rangle \\ &= \frac{1}{2} \left(\sum_{p=1}^{d} \sum_{q=1}^{d} \sum_{m=1}^{k_w} e_{mp} e_{mq} v_{mk} - \sum_{p=1}^{d} \sum_{m=1}^{k_w} e_{mp}^2 v_{mk} \right) \\ &= \frac{1}{2} \sum_{m=1}^{k_w} \left(\left(\sum_{p=1}^{d} e_{mp} \right)^2 v_{mk} - \sum_{p=1}^{d} e_{mp}^2 v_{mk} \right) \end{aligned}$$

From the above equations, we know the computation complexity of z_k is in $O(dk_w)$, and the complexity of calculating the embedding vector $\mathbf{z}$ for each case is $O(dk_w k_v)$. Besides, since only non-zero features kick in, we only need to sum over all non-zero features during the calculation. Thus, the complexity for calculating $\mathbf{z}$ is reduced to $O(d_n k_w k_v)$ where $d_n \ll d$ denotes the number of non-zero features.

9.4.5.2 Time Complexity of HeCBR

The time complexity of HeCBR mainly comes from the phase of case retrieval in practical applications, since the other two time-consuming phases, i.e., case representation and

case retention, can be done offline. In case retrieval, the time complexity consists of two parts: generating the hash code for each new case and reranking all the retrieved candidate cases, since the similar case candidates can be obtained with a time complexity of $O(1)$ in the hash table. The approximate time for generating hash code is $O(d_n k_w + d_n k_w k_v + l k_v^2 + r)$ in which the four parts correspond to the time need for feature embedding, multiview feature interaction, fully-connect layers and the hash layer in the hash network respectively. Assume we obtain n_r candidate cases, the time for reranking the candidates consists of $O(n_r d)$ for calculating the distances using the original features of cases and $O(n_r \log n_r)$ for ranking the distances if using the quick sort algorithm. Compared to the step of reranking, we can generate the hash codes for a batch of cases in parallel and even accelerate the calculation by a GPU processor. Thus, the time for generating hash codes can be ignored. Accordingly, the approximate time complexity of HeCBR is as follows: $O(n_r d + n_r \log n_r)$. Since we have $n_r \ll n$ which is usually the case and the average number of cases in each bucket is $n/2^r (\approx n_r)$, the complexity of HeCBR is far less than those CBR methods which traverse the case base, i.e., $n_r d + n_r \log n_r \ll nd + n \log n$. In addition, comparing to other clustering-based CBR methods, HeCBR obtains the candidate cases by hash table lookup with a time complexity $O(1)$ and also improves the retrieval efficiency.

9.5 Experiments and Evaluation

In this section, we conduct extensive experiments to investigate the following research problems:

Q1 How does HeCBR perform in terms of classification?

Q2 How does HeCBR perform in terms of the similarity retrieval task?

Q3 How robustly does HeCBR perform with different hyperparameters?

Q4 How does the adaptive update in case retention affect the performance of HeCBR?

9.5.1 Experimental Settings

9.5.1.1 Datasets

We verify the effectiveness of our proposed HeCBR on eight real-world high-dimensional heterogeneous datasets, which contains binary classification and multiclass classification

Table 9.1: Data characteristics of eight high-dimensional sparse datasets.

Dataset	Abbr.	#instances	#dimension	#class	sparsity
Internet Advertisements	ADV	3279	1557	2	0.01
Protein	PT	17766	357	3	0.29
Adult	ADT	45222	118	2	0.35
Dota2	Dota	102944	172	2	0.08
Character Font Images	Font	391651	896	142	0.25
Movie Tweetings	MT	773442	90191	10	5.00E-05
Criteo	CT	1000000	199	2	0.2
MovieLen-1M	ML1M	1000209	9794	5	7.00E-04

problems and covers various domains such as transaction classification, movie rating prediction, font image classification, and protein classification: (1) Internet Advertisements (short for ADV) collects a set of possible advertisements on Internet pages, and the task is to classify an image into an advertisement or not[2]. (2) Protein (PT), a multiclass classification dataset, contains protein information for studying the structure of proteins[3]. (3) Adult (ADT) collects $45,222$ census records extracted from 1994 Census database, which contains both categorical and numeric attribute for classification task[4]. (4) Dota2 (Dota) collects the battle formation from a popular computer game Data2 with two teams of 5 players, and the task is to predict which team wins. The above three datasets are collected from the UCI machine learning repository[5]. (5) Character Font Images (Font) consist of images from 153 character fonts and record a variety of font description, we select 142 of 153 character fonts in the experiments[6]. (6) Movie Tweetsing (MT) is a dataset consisting of ratings on movies that were contained in well-structured tweets on Twitter[7]. (7) Criteo (CT) includes 45 million user click records and contains both continous and categorical features[8]. Considering the computation burden, we randomly select 1 million records from the Criteo dataset for evaluation in the experiments. (8) MovieLen-1m (ML1M) collects about 1 million anonymous ratings of approximately $3,900$ movies made by $6,040$ MovieLens users[9].

The detailed characteristics of the eight datasets are reported in Table 9.1. The

[2]https://archive.ics.uci.edu/ml/datasets/Internet+Advertisements
[3]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[4]https://archive.ics.uci.edu/ml/datasets/Adult
[5]https://archive.ics.uci.edu/ml/index.php
[6]https://archive.ics.uci.edu/ml/datasets/Character+Font+Images
[7]http://github.com/sidooms/MovieTweetings
[8]https://www.kaggle.com/c/criteo-display-ad-challenge
[9]https://grouplens.org/datasets/movielens/

table shows the numbers of instances, features, classes and sparsity of each dataset where $sparsity$ reflects the proportion of non-zero features. Note that we convert the categorical features into binary features via one-hot encoding in each dataset. Table 9.1 reports the data characteristics of converted datasets.

9.5.1.2 Baselines

To investigate the performance of HeCBR, we first compare HeCBR with hashing-based methods including four state-of-the-art LSH-based CBR methods and four dimension reduction and representation methods as follows.

- LSH: It incorporates the original locality-sensitive hash algorithm [87] to map the cases into binary hash codes for efficient case retrieval.
- WTAH: It introduces WTAHash, a sparse embedding method, to CBR that transforms the input into binary codes guaranteeing Hamming distance in the resultant space closely correlated with rank similarity measures.
- FlyH: It applies FlyHash [43], a hash algorithm inspired by fruit flies' olfactory circuits, to improve the performance of case retrieval on high-dimensional data.
- PMH: It uses PM-LSH [284], a latest fast and accurate LSH framework based on a simple yet effective PM-tree, to improve the performance of computing c-ANN queries on high-dimensional data.
- ITQ: The method utilizes ITQ [61], which adopts PCA for dimension reduction and quantization, to generate binary representation for case retrieval.
- NFM: The method involves NFM [249], a deep neural factorization machine method employing feature embedding for handling high-dimensionality and sparsity, to extract case representation.
- SVAE: The method adopts SVAE [107], a sparse variational autoencoder specified to address high-dimensional and sparse data, to represent cases.
- M2V: The method introduce Mix2Vec [294], a state-of-the-art unsupervised mixed data representation based on mixed feature embedding, to obtain high-dimensional and heterogeneous case representation.

Note that NFM, SVAE, and M2V generate dense low-dimensional data representation rather than hash codes. To guarantee a fair comparison, we binarize the dense data representation to generate hash codes, where we add a new similarity-preserving objective under a joint-training manner for unsupervised representation methods, i.e., SVAE and M2V. Obviously, LSH, WTAH, FlyH, and PMH denotes the CBR baseline enabled by data-independent hash methods, while the other four denote competitors equipped with data-dependent (deep) hash methods. In addition, we compare HeCBR with the state-of-the-art CBR methods to investigate CBR performance in terms of retrieval efficiency.

- SNCBR [165]: It utilizes a heuristic simulated annealing algorithm to optimize the weight allocations in similarity calculation.

- GACBR [65]: The method adopts a genetic algorithm to optimize the feature weights to improve similarity calculation.

- MCCBR [2]: The method obtains more rational weight allocations by the predefined evolution and communication rules and a regional sub-algorithm based on SA.

- ANNCBR [13]: It trains a neural network by predicting classification labels and treats connection weights as the corresponding attribute weights.

- HCBR [277]: A state-of-the-art CBR model introduces conceptual clustering to capture structural relations among cases and incorporates the relations for calculating structural similarity.

These state-of-the-art methods are deliberately chosen for the following considerations: 1) we introduce state-of-the-art LSH into CBR methods since prior hash-based CBR models generally adopt LSH to improve retrieval efficiency; 2) we compare HeCBR with dimension reduction and representation methods to verify the superiority of HeCBR in addressing high-dimensionality and heterogeneity issues; 3) we choose the state-of-the-art CBR methods to investigate the retrieval efficiency and CBR performance of HeCBR over traditional CBR methods. To our best knowledge, few deep models are incorporated to improve the performance of CBR, let alone deep hash models. Therefore, we do not compare HeCBR with the state-of-the-art deep hashing methods.

9.5.1.3 Evaluation Measures

In the experiments, we perform 5-fold cross-validation and report the average evaluation results in the experiments. Specifically, we employ accuracy and AUC (area under the ROC curve) to evaluate classification performance. To evaluate the retrieval performance, we also adopt mean average precision (MAP) and precision@K (Prec@K).

We perform a grid search of the number of hashtables over $\{4,8,16,32,64\}$ and the bucket width over $\{5,10,15,20,25\}$ on validation sets to find the optimal configuration for the four LSH-based CBR baselines. In addition, we adopt the parameter settings recommended by the authors for the other comparative baselines. For our proposed HeCBR, we tune the embedding dimension k_w and view dimension k_v over $\{15,32,64,126,256\}$ by a grid search and adopt $k_w = 64$, $k_v = 64$, $r = 36$ and $l = 3$ for the fully-connected layers Φ_F, i.e., the shape of dimension in Φ_F is fixed as $64-128-128-r$ if not specified. In addition, we perform a grid search over $\alpha \in \{0.2,0.4,0.6,0.8\}$ and $\lambda \in \{0,0.2,0.4,0.6,0.8\}$ with a step of 0.2, to obtain the best results, and fixed $\beta = 0.5$. To facilitate fair comparison, we adopt the same parameter settings for the variants of HeCBR. In addition, for all comparative methods, we select $N = 10$ most similar cases to suggest class labels and report the best results if not specified.

9.5.2 Classification Evaluation

In this section, we investigate the performance of HeCBR in terms of case-based classification to verify the advance of HeCBR in handling high-dimensionality and heterogeneity.

Table 9.2: Comparison of classification performance in terms of accuracy and the area under the ROC curve (AUC).

Accuracy										
Dataset	LSH	WATH	FlyH	PMH	ITQ	NFM	SVAE	M2V	HeCBR†	HeCBR
ADV	0.7975	0.7996	0.7966	0.8261	0.9258	0.8589	0.8766	0.8522	0.9484	**0.967**
PT	0.385	0.3875	0.3916	0.4136	**0.5054**	0.4189	0.4408	0.4182	0.4807	0.505
ADT	0.7102	0.6793	0.7223	0.7467	0.7573	0.7621	0.7631	0.7641	0.8004	**0.8129**
Dota	0.4813	0.4956	0.4927	0.5011	0.5015	0.4993	0.5045	0.5132	0.5315	**0.5446**
Font	0.4203	0.4171	0.4207	0.4234	**0.5083**	0.4598	0.4721	0.4807	0.4941	0.5047
MT	0.2005	0.1858	0.1907	0.2031	0.1917	0.2038	0.2087	0.2026	0.2128	**0.2172**
CT	0.7233	0.7192	0.7274	0.7342	0.7352	0.7292	0.7356	0.7384	0.7413	**0.7445**
ML1M	0.2768	0.2822	0.2785	0.2821	0.2903	0.291	0.2933	0.2982	0.3089	**0.3168**
Avg.R	8.75	8.875	8.5	6.625	4.25	5.5	4.625	4.375	2.25	**1.25**
AUC										
Dataset	LSH	WATH	FlyH	PMH	ITQ	NFM	SVAE	M2V	HeCBR†	HeCBR
ADV	0.578	0.5832	0.5724	0.6032	0.8806	0.6288	0.7013	0.6877	0.907	**0.9162**
PT	0.5022	0.5167	0.5153	0.5233	**0.6512**	0.5275	0.5382	0.5276	0.6328	0.6465
ADT	0.6427	0.6198	0.6832	0.6853	0.6994	0.7213	0.7412	0.7447	0.8273	**0.8347**
Dota	0.5004	0.5061	0.5033	0.5126	0.5087	0.5016	0.5041	0.507	0.5444	**0.555**
Font	0.5177	0.509	0.511	0.5144	0.6126	0.5391	0.5402	0.5652	0.6017	**0.6163**
MT	0.5023	0.5011	0.5017	0.5033	0.5031	0.5115	0.5186	0.5163	0.5215	**0.5371**
CT	0.5248	0.5248	0.5248	0.5248	0.5248	0.5515	0.5492	0.5368	0.5763	**0.5908**
ML1M	0.5003	0.5005	0.5003	0.5103	0.5132	0.5189	0.5147	0.5219	0.5546	**0.5679**
Avg.R	8.6875	8.375	8.6875	6.65	4.375	5.5	4.5	4.875	2.25	**1.125**

9.5.2.1 Classification Performance

To verify the effectiveness of HeCBR, we compare it with several state-of-the-art LSH-based CBR methods and representation learning methods in terms of the case-based classification task[10]. The performance of all methods in terms of accuracy and AUC are reported in Table 9.2 where the best results in each row are highlighted in bold and the best baseline method is underlined for each dataset. The results are obtained with 36-bit binary codes and top-10 most similar cases. Avg.R denotes the average rank of each method over all the datasets, and HeCBR† denotes the variant of HeCBC which does not retain solved cases to update hash functions and hash codes. Table 9.2 enables the following key observations.

- First, compared with the baselines, HeCBR† and HeCBR achieve the best classification performance and rank top-2 in terms of the average rank of accuracy and AUC on all datasets. Especially, HeCBR† and HeCBR significantly improve accuracy by about 3.6% – 6.4% and AUC by more than 6.2% over the best baselines on Dota, ML1M, and ADT and obtain a desirable improvement of classification performance on the other datasets except for datasets PT and Font. All the results report that HeCBR† and HeCBR outperform the baselines in terms of the classification task.

- Second, LSH-enabled baselines perform much worse than the other comparative methods on most datasets, which verifies the superiority of data-dependent hashing methods over the data-independent ones in capturing data-specific features and addressing complex data issues, e.g., data heterogeneity. In addition, we observe the average accuracy and ACU of HeCBR† and HeCBR improve respectively by more than 9% and 17% over the LSH-enabled baselines. The results show that HeCBR outperforms state-of-the-art hash-enabled CBR methods which generally adopt hash methods from the LSH family to improve case retrieval efficiency.

- Third, comparing HeCBR† with the state-of-the-art data-dependent hash baselines, HeCBR† achieves higher average accuracy and AUC ranks on all datasets, indicating much better and more robust classification performance. Specifically, HeCBR† improves accuracy by more than 3.5% and AUC by more than 6.3% over the baselines on Dota, ML1M, and ADT. The results indicate that our proposed adaptive hashing network, especially the proposed Multiview Feature Interaction, is more

[10]Case-based classification is a common task in CBR and is convenient to evaluate the performance of CBR. To address a classification task, we adopt the majority voting to suggest the class label with the most votes (i.e., the label with the most number of supported cases) in top-N most similar ones.

effective than the baselines in handling high-dimensionality and heterogeneity and representing complex cases.

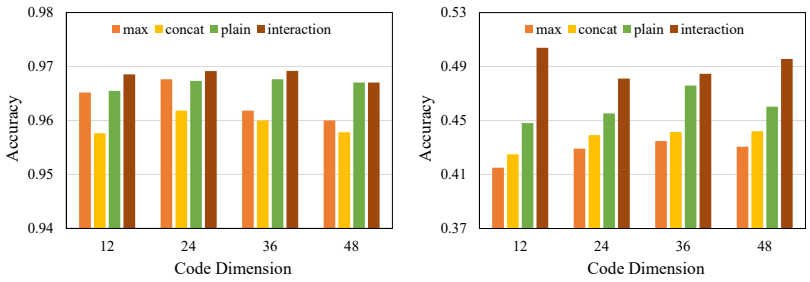
- Fourth, HeCBR† performs worse than ITQ on datasets PT and Font, which is attributable that high-dimensional datasets, i.e., PT and Font (image), have a large proportion of numeric features where PCA performs better than the feature representation methods (NFM, M2V and HeCBR) and reconstruction-based representation method (SVAE) to obtain effective and concise case representation under the high-dimensionality setting.
-
- Finally, HeCBR consistently performs better than HeCBR† and achieves obvious better accuracy and AUC than HeCBR† on all datasets. The results reflect that the incrementally retained solved cases to update hash functions is beneficial to retaining new knowledge for future problem-solving and our proposed update mechanism is effective in adaptively updating hash functions and hash codes.

9.5.2.2 Ablation Study

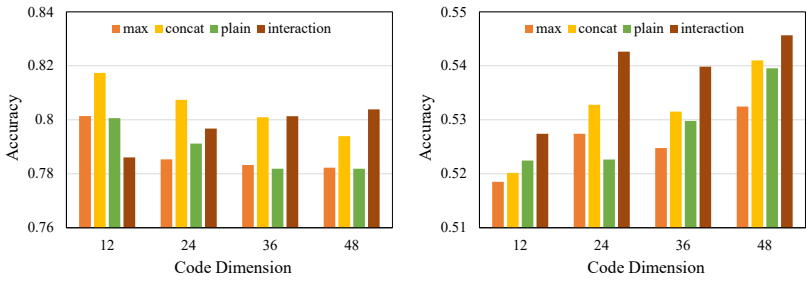
To investigate the effectiveness of the proposed feature embedding and multiview feature interaction, we introduce three variants (denoted as *max*, *concat* and *plain*) of HeCBR for ablation study which replaces the proposed feature embedding or multiview feature interaction in HeCBR with specific designs:

- *max*: The variant performs the max pooling upon the feature embedding matrix $\mathbf{E}$, i.e., $max : \mathbf{E} \rightarrow \{\max(\mathbf{E}_{1\cdot}), \cdots, \max(\mathbf{E}_{k_w\cdot})\} \in \mathbb{R}^{k_w \times 1}$.
- *concat*: The variant simply concatenates all feature embedding vectors, i.e., $concat : \mathbf{E} \rightarrow \text{concat}(\{\mathbf{e}_i, \cdots, \mathbf{e}_d\}) \in \mathbb{R}^{k_w \times d}$.
- *plain*: The variant adopts a fully-connected layer to transform an original case vector (i.e., $\mathbf{x}$) to a k_w-sized vector, i.e., $plain : \mathbf{x} \rightarrow \mathbf{w}\mathbf{x} + b \in \mathbb{R}^{k_w \times 1}$.

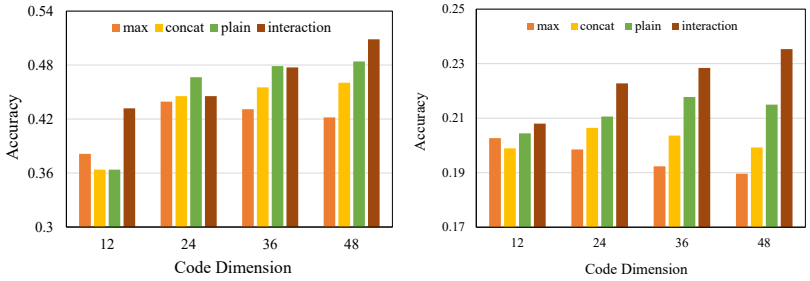
Let's denote HeCBR as *interaction* to indicate that HeCBR calculates the multiview feature interactions based on the feature embeddings. The experimental results comparing the above four methods on the eight datasets are shown in Figure 9.3, where we perform the comparative methods with $k_w = 64$ and different hash code dimensions $r \in \{12, 24, 36, 48\}$ and show the corresponding hyperparmater settings on λ and α. From the results, we obtain the following observations:



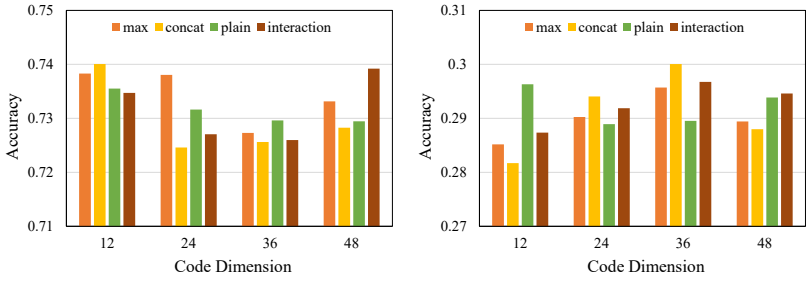
(a) ADV ($\lambda = 0.2, \alpha = 0.6$) (b) PT ($\lambda = 0.1, \alpha = 0.6$)



(c) ADT ($\lambda = 0.2, \alpha = 0.6$) (d) Dota ($\lambda = 0.0, \alpha = 0.8$)



(e) Font ($\lambda = 0.2, \alpha = 0.8$) (f) MT ($\lambda = 0.4, \alpha = 0.6$)



(g) CT ($\lambda = 0.2, \alpha = 0.6$) (h) ML1M ($\lambda = 0.2, \alpha = 0.6$)

Figure 9.3: Accuracy comparison of *interaction* and different variants under various code dimensions.

- Compared with the variants, *interaction* achieves better performance in terms of accuracy, especially on ADV, PT, Dota, Font, and MT, and catches up with the

variants as the code dimension increases on ADT, CT and ML1M. The results show the better superiority and stability of *interaction*, i.e., HeCBR, under different data characteristics.

- Specifically, *interaction* outperforms the variants *max* and *concat* on all datasets except for ADT and CT, which demonstrates the effectiveness of the proposed multiview feature interaction in capturing feature couplings. The variant *max* downsamples sharpest features, while *concat* retains all features. Their superiority on ADT and CT is attributed that they are suitable for dense data.

- Also, *plain* performs worse than *interaction* on most of the datasets, which further indicates the contribution of the proposed feature embedding for learning heterogeneity. In addition, *plain* shows competitive and even better performance than *max* and *concat* on ADV, PT, Font and MT. This is reasonable since *plain* performs direct transformation on raw features and learns effective representations from datasets with more numeric attributes, e.g., ADV, PT and Font.

In summary, *interaction* achieves much better performance, and the accuracy of variants varies largely with different datasets. The results demonstrate the effectiveness of our proposed feature embedding and multiview feature interaction in handling high-dimensionality and heterogeneity issues.

9.5.3 Retrieval Evaluation

In this section, we investigate the retrieval performance of HeCBR to verify the contributions of HeCBR to improving retrieval performance.

9.5.3.1 Retrieval Accuracy

We compare HeCBR with the state-of-the-art baselines in terms of retrieval performance. Specifically, we report the evaluation results of mean average precision (MAP) and precision on the retrieved top-N most similar cases, as shown in Figures 9.4 and 9.5 respectively. Six representative baselines are compared with HeCBR under 36-bit binary codes. From the results, we observe the following findings:

- HeCBR establishes a new state-of-the-art on all datasets except for PT, and it outperforms the baselines in terms of MAP and precision under different numbers of retrieved cases, especially on ADV, ADT, Dota and ML1M. The results show the

superiority of HeCBR over the baselines in retrieving similar cases, confirming the better classification accuracy of HeCBR over the baselines in the evaluation of classification performance.

- Compared with the baselines, HeCBR shows less fluctuation and a smooth trend in MAP and has a more stable precision along with the increasing case numbers. The results reflect that the introduction of feature embeddings and feature interactions in HeCBR is effective to capture the intrinsic heterogeneity in (high-dimensional) cases and improve the robustness of HeCBR on data of different scales and types.

- In addition, data-independent methods, i.e., the LSH-based methods and ITQ, are more vulnerable to the number of retrieved case numbers and generally perform worse than data-dependent hashing methods, for example ITQ and PMH on ADT, FlyH on Dota, and LSH and PMH on MT. The results are attributed that data-independent methods rely on specific distance measurement or transformation (PCA) and hardly learn data specific features to address complex data issues, e.g., heterogeneity and high dimensionality.

From the above results, we conclude that HeCBR effectively captures feature heterogeneity and interactions to improve case representation and performs more accurate similar case retrieval.

9.5.3.2 Retrieval Efficiency

Since HeCBR and the comparative LSH-based CBR and representation methods perform the same two-step retrieval process, i.e., retrieving candidates from hash tables and reranking the candidates for top-N similar cases, we compare HeCBR with five state-of-the-art case-based methods to further investigate the retrieval efficiency of HeCBR. Note that HCBR proposes an efficient retrieval algorithm, while the other four baselines adopt the same linear nearest neighbor search (NNS) algorithm. Therefore, we report the time cost of HeCBR, HCBR and ANNCBR in the retrieval efficiency comparison, where ANNCBR is representative for the four NNS-based baselines.

The efficiency comparison is reported in Figure 9.6 along with the classification accuracy comparison. Classification accuracy results and retrieval time costs (seconds/per 10 retrievals) are reported on the right and left of each figure respectively. From the figures, we observe that HeCBR performs worse than the CBR baselines in terms of accuracy, while it achieves much higher efficiency than the baselines. Specifically, HCBR

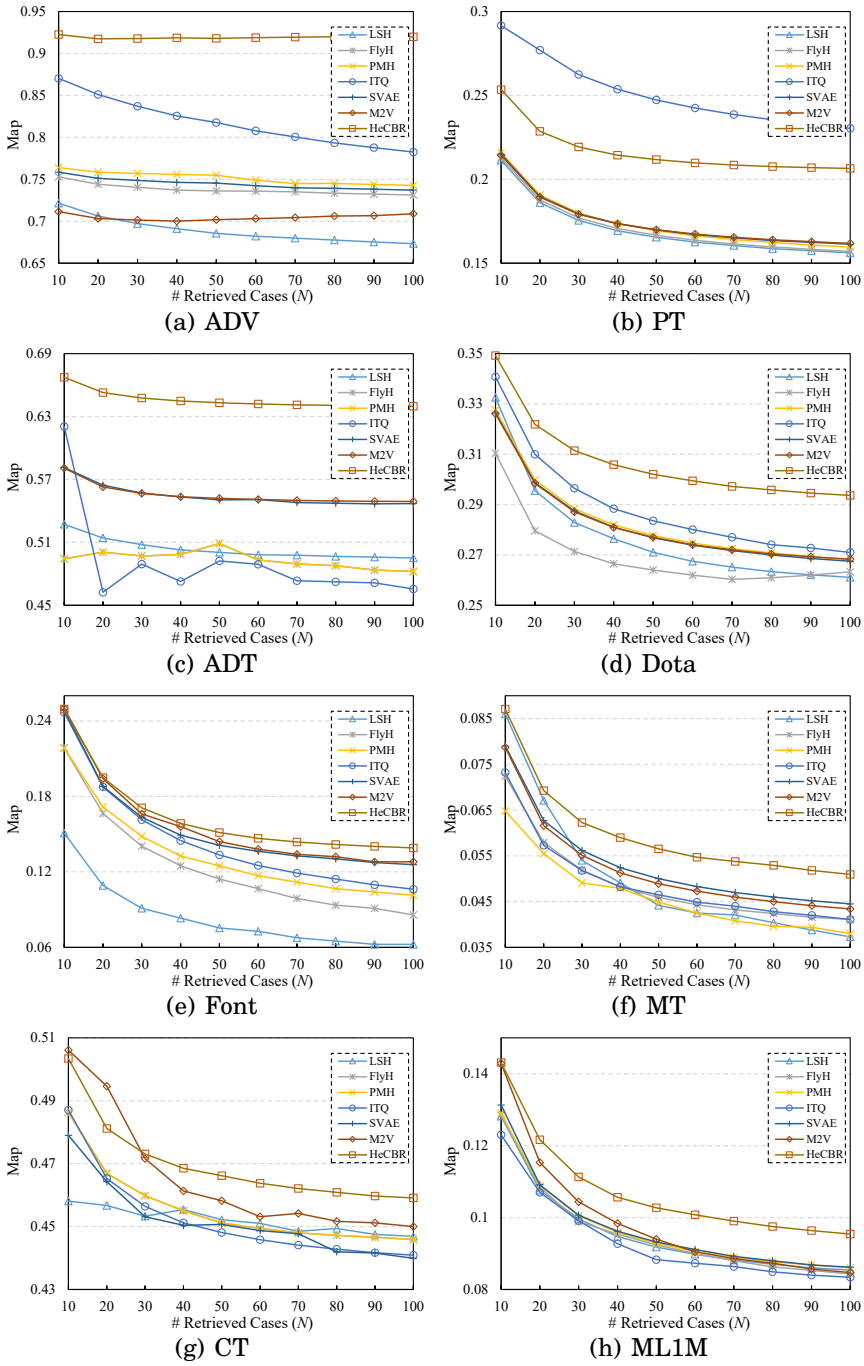


Figure 9.4: Comparison of retrieval performance in terms of mean average precision with different numbers of retrieved cases (MAP@N).

has the highest accuracy on all datasets, which is reasonable since HCBR leverages the structural information among cases to effectively optimize case similarity measures. HeCBR achieves comparable and even better accuracy compared with the other baselines

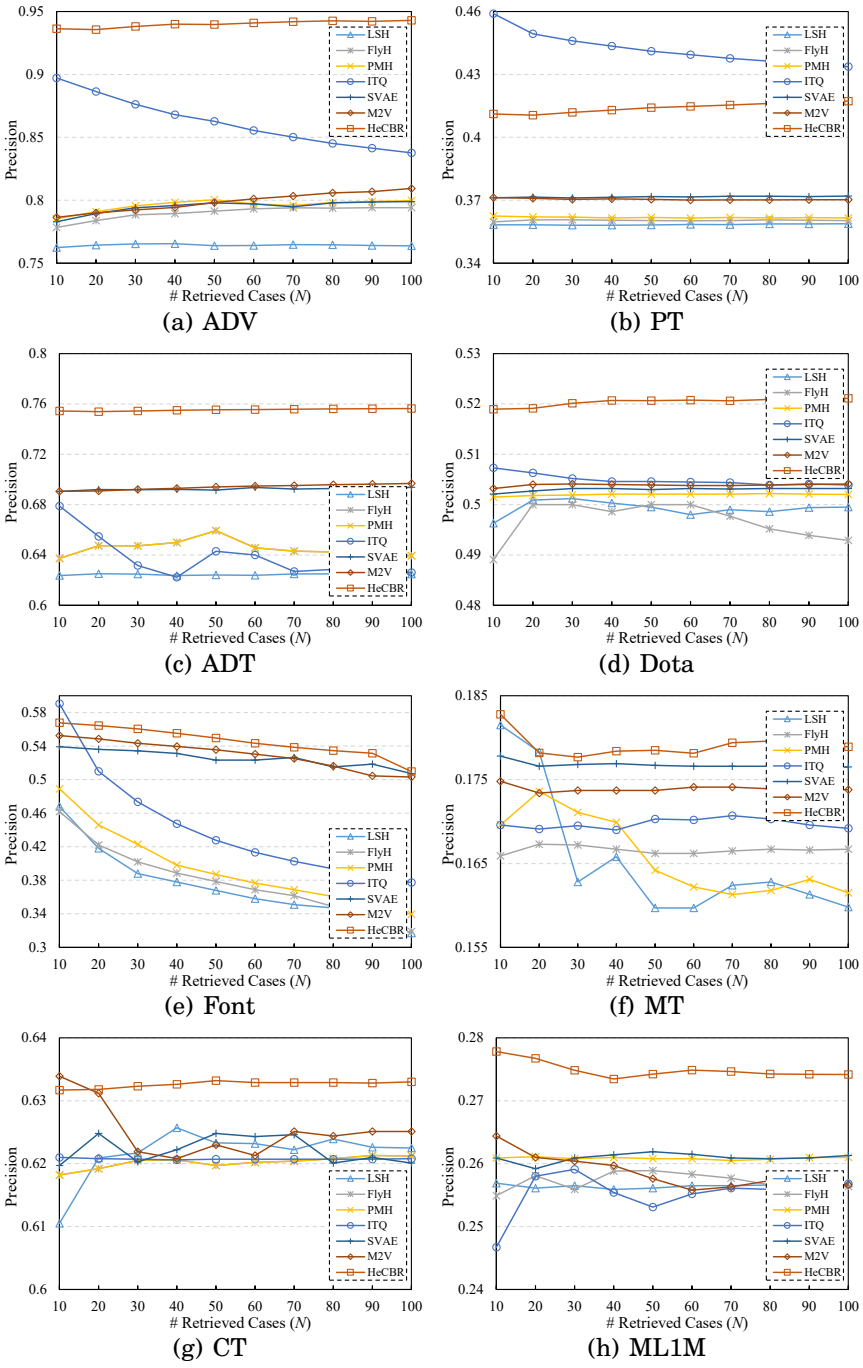


Figure 9.5: Comparison of retrieval performance in terms of precision with different numbers of retrieved cases N (Precision@N).

on most of the datasets, especially on ADT and Dota. Although HeCBR loses accuracy during learning the discrete hash codes, it enhances case representation to compensate for the loss by introducing heterogeneous embedding and capturing feature interaction.

Table 9.3: Average accuracy comparison of HeCBR under different values of the weight parameter λ and scaling parameter α respectively.

Dataset	Weight Parameter (λ)					Scaling Parameter (α)			
	0	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
ADV	0.9661	0.9585	0.9508	0.9261	0.9102	0.9655	0.9661	**0.967**	0.9652
PT	0.433	0.4677	0.4461	0.4544	0.4469	0.4718	**0.5046**	0.4785	0.4866
ADT	**0.8097**	0.7869	0.7808	0.7562	0.7586	0.7972	0.8063	0.7991	0.8033
Dota	0.5378	0.503	0.5044	0.502	0.4934	**0.5464**	0.5391	0.5429	0.5334
Font	0.5031	0.5047	0.4963	0.4834	0.4692	0.4872	0.4934	**0.505**	0.5013
MT	**0.2172**	0.2048	0.1964	0.2005	0.1889	0.1976	0.1948	0.2041	0.1889
CT	0.7383	0.7403	**0.7436**	0.7436	0.7445	0.7433	0.7273	0.7346	0.7308
ML1M	**0.3145**	0.2903	0.2817	0.2825	0.2855	0.2885	0.2898	0.2916	0.2867

In addition, HeCBR greatly improves the retrieval efficiency on all datasets such that it reduces 41% ~ 78% retrieval time costs over ANNCBR and up to 62% ~ 84% retrieval time costs over HCBR as shown on the right of each subfigure in Figure 9.6. As known, the time complexity of the NNS algorithm is proportional to the cardinality of the retrieval set, i.e., the number of cases, HeCBR thus performs increasingly efficiently over ANNCBR with the increase of the number of instances (cases) from ADV to ML1M. Regarding HCBR, it achieves desirable efficiency on MT, CT and ML1M, which is attributed that HCBR structurally organizes all cases and performs large-scale pruning, suitable for large-scale and sparse datasets. From the results, we conclude that HeCBR achieves desirable retrieval accuracy over the CBR-based based, but it greatly reduces retrieval costs. The slight sacrifice of retrieval accuracy for large efficiency improvement is considerably acceptable, especially for the online or real-time scenarios with high demands of retrieval efficiency.

9.5.4 Hyperparameter Study

To investigate the parameter sensitivity of HeCBR, we further evaluate the classification accuracy of HeCBR in terms of the weight parameter λ, the scaling parameter α, the view dimension k_v, and the embedding dimension k_w. All the experiments are conducted under the settings: $\lambda = 0.2$, $\alpha = 0.6$, $k_v = 64$ and $k_w = 64$ if not specified.

9.5.4.1 Evaluating HeCBR w.r.t. Different α and λ

As shown in Table 9.3, we perform a grid search on λ over $\{0, 0.2, 0.4, 0.6, 0.8\}$ and α over $\{0.2, 0.4, 0.6, 0.8\}$ and report the average accuracy under each value of λ and α

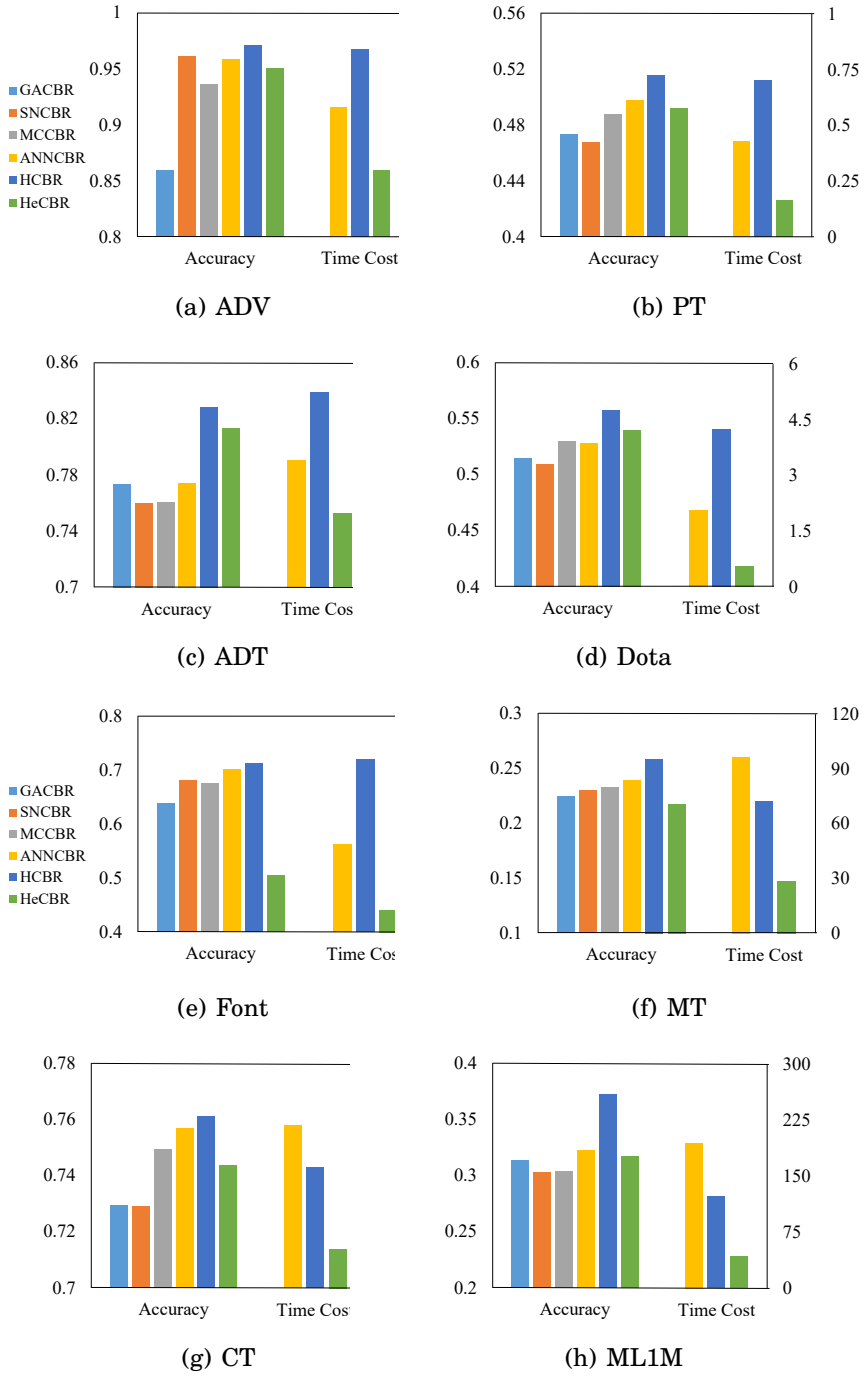


Figure 9.6: Accuracy and efficiency of comparison with state-of-the-art case-based classification methods.

respectively. The lowest average accuracy values among λ and α on each dataset are underlined respectively, and the highest average accuracy value on each dataset is highlighted in bold. From the table, we observe that: 1) HeCBR obtains relatively higher

performance at $\lambda = 0.2$ and $\alpha = 0.6$ respectively since HeCBR does not hit the lowest accuracy at $\lambda = 0.2$ or $\alpha = 0.6$ and achieves much better accuracy compared with other settings. 2) In contrast to α, HeCBR has larger but more desirable accuracy fluctuation on λ, indicating the necessity of a more careful selection of λ than that of α. This is reasonable because the weight parameter λ largely influences the learning objective while α mainly scales the similarities to guarantee higher gradients from the Sigmoid function during backpropagation. 3) In addition, with the increase of λ, HeCBR may have an increase of accuracy but then has a great decrease when λ reaches larger values, e.g., 0.8, which indicates large λ may mislead the learning objective and suggests a relatively small value of λ. On the contrary, HeCBR achieves better performance at $\alpha = 0.4$ or $\alpha = 0.6$, suggesting a moderate value of α. The results show that large α may not work as a scaling parameter while small α may excessively erase the similarity difference.

9.5.4.2 Evaluating HeCBR w.r.t. Different View Dimension k_v and Embedding Dimension k_w

We perform a grid search on k_v over $\{16, 32, 64, 128, 256\}$ and k_w over $\{16, 32, 64, 128, 256\}$ and report the average accuracy under each value of k_v and k_w respectively in Table 9.4. The lowest average accuracy values between k_v and k_w on each dataset are underlined respectively, and the highest average accuracy value on each dataset is highlighted in bold. From the results, we observe that: 1) HeCBR achieves slightly better performance when k_v and k_w take more moderate values, e.g., $k_v, k_v \in \{32, 64, 128\}$, while smaller or larger embedding dimensions may lead to more inferior performance possibly due to underfitting and overfitting respectively; 2) overall, HeCBR achieves stable and desirable performance, e.g., without terribly poor accuracy, on all the datasets over the grid search.

9.5.5 Performance Under Adaptive Update

To investigate the stability of HeCBR under different training sample rates, we gradually increase the proportion of training samples on each dataset in Table 9.1 from 10% to 90% in the experiments. We compare HeCBR with its variant (denoted as **w/o update**) that does not apply the adaptive update strategy to retain newly-solved cases in terms of classification accuracy, and the results are shown in Figure 9.7, which indicate three observations:

- HeCBR outperforms its variant **w/o update** nearly under all sampling rates on all datasets, indicating that the update strategy effectively retains beneficial cases to

Table 9.4: Average accuracy comparison of HeCBR under different values of the view dimension k_v and embedding dimension k_w respectively.

Dataset	View Dimension					Embedding Dimension				
	16	32	64	128	256	16	32	64	128	256
ADV	0.9685	0.9673	0.9664	**0.9698**	0.9682	0.9658	0.967	0.9658	0.9634	0.9652
PT	0.433	0.492	0.446	0.454	0.447	0.472	**0.505**	0.492	0.487	0.485
ADT	0.7972	0.8049	0.8102	**0.8114**	0.8079	0.7891	0.7937	0.7963	0.7998	0.8003
Dota	0.5308	0.5353	**0.5458**	0.5341	0.5421	0.5375	0.5446	0.5413	0.5396	0.5313
Font	0.5023	0.5047	0.4976	0.4821	0.4687	0.4867	0.4953	**0.5049**	0.5041	0.4943
MT	0.2037	0.1955	0.1951	0.1971	0.1839	0.2005	**0.2061**	0.1996	0.2006	0.1969
CT	0.7346	0.7375	0.7339	0.7326	0.7308	0.7352	0.7386	**0.7393**	0.7364	0.7361
ML1M	**0.3253**	0.2945	0.2969	0.2936	0.2919	0.2855	0.294	0.2938	0.2768	0.287

update the case base and hash function for improving future classification.

- With the increase of training sample rates, HeCBR shows less fluctuation and increase than the variant **w/o update**, demonstrating the robustness of HeCBR in relation to different proportions of training samples. The large accuracy increase of **w/o update** is attributed to the fact that more samples generally benefit the performance even for non-incremental methods.

- When the training sample rates reach 80% 90%, the variant **w/o update** gets comparable or slightly better accuracy than HeCBR. This is because HeCBR may lose its superiority or even introduce noises with the update strategy when the training samples are sufficient. Notably, the proposed adaptive strategy effectively avoids invalid updates and alleviates the inferior performance to a great extent.

9.6 Conclusions

In this chapter, we focus on the practicality of learning to hash approaches on real-world, especially high-dimensional and heterogeneous case data. In summary, the contributions of this work mainly include:

- We propose an adaptive hashing network to build a hashing-enabled CBR system. The proposed hashing network learns similarity-preserving compact hash codes of the cases and the corresponding hash function simultaneously. It is beneficial to map the high-dimensional cases into the low-dimensional hash space to reduce storage cost and approximate the nearest neighbor search in the hash space to improve similarity retrieval efficiency.

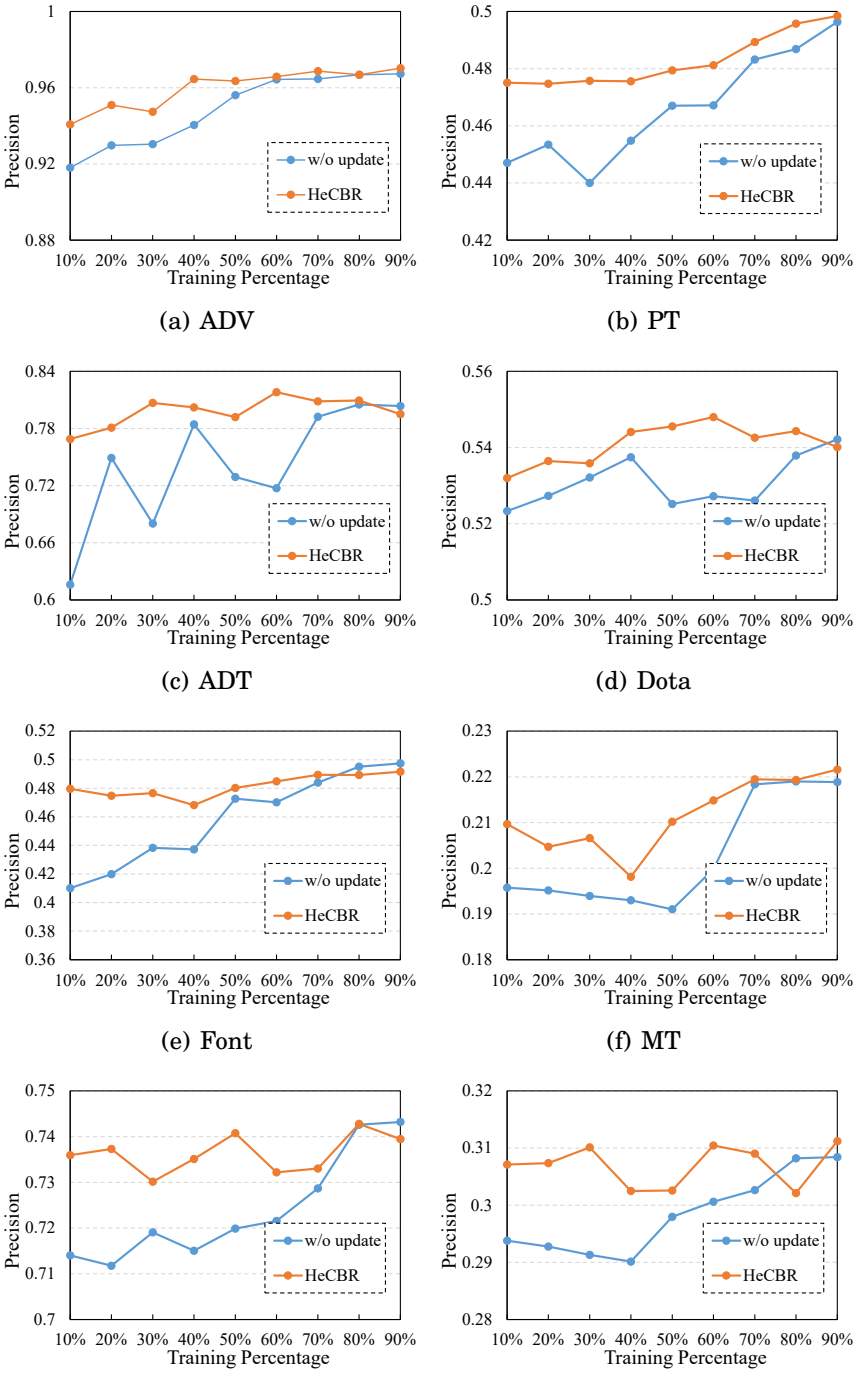


Figure 9.7: Accuracy comparison of HeCBR and its variant **w/o update** under different training sample rates.

- To tackle the challenges and gaps of high-dimensional heterogeneous case data, we introduce position embedding to perform feature embedding and propose a

multilinear interaction layer to represent complex cases. The network design filters out all zero-valued features and captures the feature couplings, assisting in efficiently handling high-dimensionality and heterogeneity while preserving similarity relationships.

- We further introduce an incremental learning mechanism to retain the solved cases. Specifically, we propose an adaptive learning objective to update the hash function and an updating strategy to efficiently update the hash codes of all cases.

We perform a collection of experiments on eight real-world datasets of different applications and make comparisons with other state-of-the-art hashing-enabled CBR models and several typical CBR models to investigate the effectiveness of our proposed HeCBR. All experimental results demonstrate that our HeCBR significantly outperforms the hashing-enabled methods in terms of classification tasks and achieves desirable performance compared with typical CBR methods. Theoretical and empirical analysis show that HeCBR can greatly reduce storage cost and significantly improve efficiency in relation to typical CBR models.

Part V

Summary and Prospect

CHAPTER 10

CONCLUSION

In this thesis, we analyzed the issues and challenges in modeling non-IIDness and in non-IID applications. We conclude the necessity and benefits of building non-IID data learning models that consider heterogeneities and coupling relationships in various real-world data and different application scenarios. Subsequently, we presented the design of six non-IID modelings in terms of different real-world applications, i.e., recommender systems, multivariate time series forecasting, and learning to hash. In this chapter, we conclude the contributions of this thesis.

10.1 Non-IID Recommender Systems

In Part II, we focus on designing non-IID recommender systems via modeling the user/item/context feature heterogeneity and capturing their underlying coupling relationships and also study to debias rating estimate via exploring the rating generation process to address the different rating distributions. We build two non-IID modelings of two typical RSs: sequential RSs and collaborative filtering.

10.1.1 Sequential Recommendation

In Chapter 4, to understand how the explicit features of users, items and context influence user preferences and how user preference dynamics influence user actions over time, we propose a time-aware recommendation neural network, TARN. TARN learns the

couplings between explicit user features, item features, and time-related contextual information by a feature interaction network and models the temporal dynamics of user preferences by a convolutional network. To enhance the model's capability of preference modeling, we further propose a temporal action embedding and attentive average pooling.

Extensive experiments verify the advantages of TARN and the effectiveness of integrating feature couplings and preference dynamics in capturing intrinsic driving factors of recommendation. Significantly, TARN provides a way to explain user stationary preferences in terms of explicit feature couplings. The visualization results support the rationale of our design and show interesting findings that inter-feature couplings between users, items and context are more highly contributive than the intra-feature couplings (i.e., first-order couplings within user features, item features or context features).

10.1.2 Collaborative Filtering

In Chapter 5, we propose a new framework TCF to model the missing-not-at-random rating generation and estimate the MNAR ratings by deeply exploring the relations between rating missingness, item observability, and user selection. The proposed framework includes three sub-models for jointly inferring triple aspects: item observability, user selection and ratings. The newly-added latent variable observability distributes a confidence of being truly negative to each missing entry. We also instantiate the framework to a probabilistic model TPMF, which further introduces the factor dependency between user selection and ratings to model their multifaceted factor correlation. Extensive experiments on the synthetic datasets show that TPMF effectively model the triple aspects simultaneously and infer their relationships.

Results on real-world datasets show that both item observability and factor dependency are critical to MNAR rating estimation and TPMF outperforms the state-of-the-art debiasing methods in rating prediction with respect to RMSE and MAE. Further work includes introducing extra metadata into modeling item observability, which may improve the estimate accuracy of item observability and alleviate overfitting issues and even cold-start issues.

10.2 Non-IID Multivariate Time Series Forecasting

In Part III, we focus on studying the non-IIDness in multivariate time series data and explore the inter-/intra-series couplings and the heterogeneity of time series variables. We

deliver two neural models based on Transformers and deep cross network to successful address the non-IIDness in multivariate time series.

10.2.1 Transformer-based Model

In Chapter 5, we propose a novel spectral clustering-enhanced neural sequential MTS forecasting model Cospectrumer. The model takes advantage of a spectral clustering network and a clusterwise forecasting network. The spectral network combines discrete Fourier transform and neural networks and adopts the spectral relaxation of K-means objective to learn a cluster indicator matrix and generate clusterwise MTS inputs. The forecasting network introduces a heterogeneous embedding module to capture the heterogeneity of mixed inputs in MTS and adopts Transformers as the backbone and the encoder-decoder architecture to model MTS sequences. The network stacks multiple Transformers and proposes a cluster-aware multi-head attention mechanism to capture information attention among different clusterwise inputs. Overall, the clustering network divides and correlates heterogeneous time series in order to model inter-series couplings, while the forecasting network aims to capture both intra-series temporal patterns and inter-series correlations to improve the accuracy of MTS forecasting.

Extensive experiments on four datasets verify that Cospectrumer outperforms the SOTA baselines and achieves better MTS forecasting performance. Cospectrumer has relatively high model complexity compared with other SOTA Transformer-based models and relies on existing efficient self-attention mechanisms to improve its efficiency.

10.2.2 Deep Coupling Network

In Chapter 7, to address the relationship modeling issues for MTS forecasting, we propose a novel model DeepCN for multivariate time series forecasting, including single-step forward prediction and multi-step forward prediction. Compared with the previous work, in this section we build our model based on the couplings which can bring more comprehensive information to enhance representations of relationships among time series. Specifically, first we revisit the relationships among time series from the perspective of mutual information. Then based on the analysis, we design a coupling mechanism to learn the hierarchical and diverse couplings to represent the relationships which can comprehensively exploit the intra- and inter-series dependencies. The coupling mechanism can model multi-order couplings and account for time lag effect explicitly. After that, since different variables exhibit different patterns, we leverage a coupled variable

representation module to learn the variable relationship representation. Finally, we make predictions by one forward step which can avoid error accumulation. The one forward step method makes our model more efficient and stable in the multi-step forward prediction.

We conduct extensive experiments on five real-world datasets and compare our model with other state-of-the-art baselines. The experimental results show that our model achieves superior performances. Furthermore, more analysis about our model demonstrates that different order couplings impose different effects on different datasets which give us enlightenment for handling different types of MTS data.

10.3 Non-IID Learning to Hash

In Part IX, we focus on the influence of non-IIDness, precisely the couplings between inputs and hash codes and data heterogeneity, to learning to hash modeling and applications. Specifically, we first introduce the coupling model to enhance the informativeness and compactness of hash codes and then investigate its effectiveness on real-world heterogeneous case-based retrieval and classification.

10.3.1 Hash Quality

In Chapter 8, to improve the quality of hash codes in deep supervised hashing, we propose a novel probabilistic code balance constraint, which forces each hash bit to independently satisfy a discrete uniform distribution on $\{-1,1\}^K$ (K is hash code length). We prove and analyze the effectiveness and insights of the proposed constraint, and further incorporate the Wasserstein regularization to implement the constraint.

Extensive experiments by comparing six DSH baselines and their WR-enabled variants on two image benchmark datasets demonstrate the probabilistic code balance can effectively promote retrieval performance and improve the compactness and informativeness of hash codes.

10.3.2 Case-based Retrieval and Classification

In Chapter 9, we propose a novel deep hashing network to enhance case-based reasoning. Specifically, the proposed network introduces multiview feature interactions to represent high-dimensional and heterogeneous cases and generates binary hash codes with a quantization regularizer to control the quantization loss. We further propose an adaptive

learning loss to strategically update the hash function in the phase of case retraining. Extensive experimental results on public datasets show the superiority of HeCBR over the state-of-the-art hash-based CBR methods in terms of classification and retrieval performance and demonstrate the higher efficiency of HeCBR than the state-of-the-art CBR methods.

CHAPTER 11

OPEN CHALLENGES AND FUTURE DIRECTIONS

At the current stage, research on practical non-IID learning approaches still encounters numerous challenges. Although some typical issues and problems have not been addressed and solved, there are multiple open issues necessary to be considered to make current non-IID approaches better to qualify real-world complex scenarios.

11.1 Quantification and Evaluation Methods

Open Challenges

- How can we well quantify the non-IIDness of real-world data?
- How can we properly evaluate the non-IIDness modeling in non-IID learning approaches?

Although there are various evaluation methods and metrics for different data learning tasks, e.g., AUC and accuracy for classification, RMSE and MAR for forecasting, and MAP, recall and precision for recommendation and information retrieval. There are few researchers focusing on measuring the heterogeneity and coupling relationship to quantify the non-IIDness of real-world data. It is also necessary and meaningful to evaluate how well non-IID approaches capture the non-IIDness, i.e., heterogeneity and couplings.

Future Directions

- Heterogeneity Quantification
  Heterogeneity must be quantified to better indicate the characteristics or complexity of real-world data, which would help the development of non-IID modeling, framework, and approaches. Although there are a few studies characterizing heterogeneity quantification, it lacks formal or widely adopted formulations to quantify heterogeneity, especially considering complex properties, aspects, and subjects associated with heterogeneity [22]. It is promising and contributive to study the quantification of heterogeneity to promote non-IID learning.

- Coupling Quantification
  Similarly, the coupling relationship must be characterized and quantified. Previous work has attempted to define and formulate related concepts, e.g., dependency and correlation. Since coupling is a much richer concept than dependency and correlation, quantifying coupling is more challenging but significant to the data learning community. We need to clarify the granularity of couplings, understand coupling taxonomy, and acquire their domain knowledge for specific practical scenarios.

- Evaluation metrics for Non-IIDness Modeling
  Based on the quantification of non-IIDness, i.e., heterogeneity and coupling, we can devise metrics to evaluate how well a non-IID approach captures the non-IIDness of real-world data. This may facilitate comprehensively measuring model capacity and provide interpretability of data learning models in a non-IIDness modeling perspective.

11.2 Model Complexity and Efficiency

Open Challenges

- How can non-IID learning approaches reduce model complexity and improve more efficiency?

- How can non-IID learning approaches infer efficiently model parameters?

Current non-IID learning approaches may seriously suffer from high model complexity and low model efficiency since those approaches include extra heterogeneity modeling or

coupling modeling which are usually time-consuming and complicated. Besides, non-IID learning approaches generally have complex and inefficient model inference. It is a great challenge that we propose lightweight and efficient non-IID learning models with efficient more inference.

Future Directions

- Efficient Non-IID Learning/Inference Paradigm
  Both traditional data statistics and current deep neural networks may be unsuitable to develop high-efficient non-IID learning models due to their complicated data modeling or large parameter volumes. We may borrow computation paradigms from other disciplines, e.g., adopting discrete Fourier transform in signal processing to represent sequences, or devise a new efficient non-IID learning/inference paradigm. This is critical to avoid research to consider the effect-efficiency balance, which may lead to overlooking the non-IIDness of real-world data.

- Non-IID model Compression
  Model compression is a hot topic to reduce model complexity, especially parameter volumes while preserving model capacity. It may be a good alternative to address the complexity and efficiency issues of non-IID learning models. It is promising to introduce model compression methodology into non-IID modeling and also challenging to maintain non-IIDness modeling along with model pruning.

11.3 Temporal Dynamics

Open Challenges

- How to model heterogeneity dynamics over time?

- How to model the dynamics of coupling relationship over time?

Real-world data is changing over time where its characteristic, e.g., distribution and scales, varies with time. Accordingly, the non-IIDness of real-world data is changing over time. This brings challenges to previous non-IID models that are built on static data and performs worse over time. How to capture dynamic non-IIDness, i.e., dynamic heterogeneity and dynamic couplings, and build continual non-IID learning approaches is significant and promising for the non-IIDness learning community.

Future Directions

- Dynamic Non-IIDness Modeling
  We can build time-aware non-IID learning models to cater for the dynamic characteristics of real-world data. For example, we can consider the temporal context/environment associated with real-world data to enhance non-IID learning models. Hereby, we can introduce time-wrap models, e.g., neural networks and kernel functions, to achieve non-IID modeling.

- Continual Non-IIDness Modeling
  Besides, we can also build continual non-IID learning models to continuously capture and learn the non-IIDness of real-world data. Under continual learning, non-IID models are continuously updated based on new data, e.g., the latest data, which facilities the models to capture the data dynamics (including dynamic no-IIDness) and guarantees the models are competent for future problem-solving.

11.4 Practical Scenarios

Open Challenges

- How can non-IID learning approaches be applied to other practical scenarios?

In this thesis, we study non-IID learning approaches in terms of three practical applications: recommender systems, multivariate time series forecasting, and learning to hash. For other complex scenarios, non-IIDness is urgent to be addressed to approach the nature of real-world data, and different scenarios possess different kinds of data that has different non-IIDness. How to apply existing non-IID learning approaches to new practical scenarios and how to tailor specific non-IID modeling for the new scenarios are attractive and valuable research problems. Here, we list two exemplary directions for trustworthy learning.

Future Directions

- Non-IID Federated Learning Federated learning has been an effective learning paradigm to address data security issues. It allows models trained based on distributed data, where it usually encounters data heterogeneity issues. Non-IID federated learning necessarily considers the non-IIDnesses within and between

local sources and within and between local and global sources. This facilitates to devising practical and superior federated learning models.

- Non-IID Privacy computing Another important branch of trustworthy learning/-model is privacy computing. Privacy computing protects data from disclosure by encrypting data or adding random noise. For non-IID data, it needs to consider non-IIDness to enhance privacy computing approaches, avoiding encryption or random noise distorting the inter-relationships or heterogeneity properties of real-world data.

BIBLIOGRAPHY

[1] LONGBING CAO, *Non-iid recommender systems: A review and framework of recommendation paradigm shifting*, Engineering, 2 (2016), pp. 212–224.

[2] AIJUN YAN AND HONGSHAN SHAO AND ZHEN GUO, *Weight optimization for case-based reasoning using membrane computing*, Inf. Sci., 287 (2014), pp. 109–120.

[3] M. APTE, S. VAISHAMPAYAN, AND G. K. PALSHIKAR, *Detection of causally anomalous time-series*, Int. J. Data Sci. Anal., 11 (2021), pp. 141–153.

[4] S. ARORA, M. SHARMA, AND P. ANAND, *A novel chaotic interior search algorithm for global optimization and feature selection*, Applied Artificial Intelligence, 34 (2020), pp. 292–328.

[5] L. J. BA, J. R. KIROS, AND G. E. HINTON, *Layer normalization*, CoRR, abs/1607.06450 (2016).

[6] L. BAI, L. YAO, C. LI, X. WANG, AND C. WANG, *Adaptive graph convolutional recurrent network for traffic forecasting*, in NeurIPS, 2020.

[7] S. BAI, J. Z. KOLTER, AND V. KOLTUN, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*, CoRR, abs/1803.01271 (2018).

[8] E. BEAUXIS-AUSSALET AND L. HARDMAN, *Extended methods to handle classification biases*, in DSAA'2017, 2017, pp. 765–774.

[9] S. BEGUM, M. U. AHMED, P. FUNK, N. XIONG, AND M. FOLKE, *Case-based reasoning systems in the health sciences: A survey of recent trends and developments*, IEEE Trans. Systems, Man, and Cybernetics, Part C, 41 (2011), pp. 421–434.

[10] M. I. BELGHAZI, A. BARATIN, S. RAJESWAR, S. OZAIR, Y. BENGIO, R. D. HJELM, AND A. C. COURVILLE, *Mutual information neural estimation*, in ICML, vol. 80, 2018, pp. 530–539.

[11] A. L. BERTOZZI, E. FRANCO, G. MOHLER, M. B. SHORT, AND D. SLEDGE, *The challenges of modeling and forecasting the spread of covid-19*, Proceedings of the National Academy of Sciences, 117 (2020), pp. 16732–16738.

[12] A. BEUTEL, P. COVINGTON, S. JAIN, C. XU, J. LI, V. GATTO, AND E. H. CHI, *Latent cross: Making use of context in recurrent recommender systems*, in WSDM, 2018, pp. 46–54.

[13] S. K. BISWAS, N. SINHA, B. PURAKAYASTHA, AND L. MARBANIANG, *Hybrid expert system using case based reasoning and neural network for classification*, Biol. Inspir. Cogn. Arc., 9 (2014), pp. 57–70.

[14] A. BOROVYKH, S. BOHTE, AND C. W. OOSTERLEE, *Conditional time series forecasting with convolutional neural networks*, arXiv, (2017).

[15] H. CAESAR, V. BANKITI, A. H. LANG, S. VORA, V. E. LIONG, Q. XU, A. KRISHNAN, Y. PAN, G. BALDAN, AND O. BEIJBOM, *nuscenes: A multimodal dataset for autonomous driving*, in CVPR, Computer Vision Foundation / IEEE, 2020, pp. 11618–11628.

[16] F. ÇAKIR AND S. SCLAROFF, *Adaptive hashing for fast similarity search*, in ICCV, 2015, pp. 1044–1052.

[17] D. CAO, Y. WANG, J. DUAN, C. ZHANG, X. ZHU, C. HUANG, Y. TONG, B. XU, J. BAI, J. TONG, AND Q. ZHANG, *Spectral temporal graph neural network for multivariate time-series forecasting*, in NeurIPS, 2020.

[18] D. CAO, Y. WANG, J. DUAN, C. ZHANG, X. ZHU, C. HUANG, Y. TONG, B. XU, AND ET AL., *Spectral temporal graph neural network for multivariate time-series forecasting*, in NeurIPS, 2020.

[19] L. CAO, *Non-iidness learning in behavioral and social data*, Comput. J., 57 (2014), pp. 1358–1370.

[20] L. CAO, *AI in finance: Challenges, techniques, and opportunities*, ACM Comput. Surv., 55 (2022).

[21] L. CAO, *Beyond i.i.d.: Non-iid thinking, informatics, and learning*, IEEE Intelligent Systems, 37 (2022), pp. 5–17.

[22] L. CAO, *Beyond i.i.d.: Non-iid thinking, informatics, and learning*, IEEE Intell. Syst., 37 (2022), pp. 5–17.

[23] L. CAO AND Q. LIU, *COVID-19 modeling: A review*, medRxiv, (2021), pp. 1–103.

[24] L. CAO, Q. YANG, AND P. S. YU, *Data science and AI in fintech: an overview*, Int. J. Data Sci. Anal., 12 (2021), pp. 81–99.

[25] W. CAO, L. HU, AND L. CAO, *Deep modeling complex couplings within financial markets*, in AAAI, AAAI Press, 2015, pp. 2518–2524.

[26] Y. CAO, M. LONG, B. LIU, AND J. WANG, *Deep cauchy hashing for hamming space retrieval*, in CVPR, 2018, pp. 1229–1237.

[27] Z. CAO, M. LONG, J. WANG, AND P. S. YU, *Hashnet: Deep learning to hash by continuation*, in ICCV, 2017, pp. 5609–5618.

[28] F. T. S. CHAN, *Application of a hybrid case-based reasoning approach in electroplating industry*, Expert Syst. Appl., 29 (2005), pp. 121–130.

[29] J. CHEN, C. WANG, M. ESTER, Q. SHI, Y. FENG, AND C. CHEN, *Social recommendation with missing not at random data*, in IEEE ICDM, 2018, pp. 29–38.

[30] X. CHEN, H. CHEN, H. XU, Y. ZHANG, Y. CAO, Z. QIN, AND H. ZHA, *Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation*, in SIGIR, ACM, 2019, pp. 765–774.

[31] X. CHEN AND L. SUN, *Bayesian temporal factorization for multidimensional time series prediction*, IEEE Trans. Pattern Anal. Mach. Intell., (2021), pp. 1–1.

[32] X. CHEN, H. XU, Y. ZHANG, J. TANG, Y. CAO, Z. QIN, AND H. ZHA, *Sequential recommendation with user memory networks*, in WSDM, ACM, 2018, pp. 108–116.

[33] Y. CHEN, Z. LAI, Y. DING, K. LIN, AND W. K. WONG, *Deep supervised hashing with anchor graph*, in ICCV, 2019, pp. 9795–9803.

[34] Y. CHEN, L. LI, L. YU, A. E. KHOLY, F. AHMED, Z. GAN, Y. CHENG, AND J. LIU, *UNITER: universal image-text representation learning*, in ECCV (30), vol. 12375, 2020, pp. 104–120.

[35] H. CHENG, L. KOC, J. HARMSEN, T. SHAKED, T. CHANDRA, H. ARADHYE, G. ANDERSON, G. CORRADO, W. CHAI, M. ISPIR, R. ANIL, Z. HAQUE, L. HONG, V. JAIN, X. LIU, AND H. SHAH, *Wide & deep learning for recommender systems*, in Proc. DLRS@RecSys, 2016, pp. 7–10.

[36] H.-T. CHENG, L. KOC, J. HARMSEN, T. SHAKED, T. CHANDRA, H. ARADHYE, G. ANDERSON, G. CORRADO, W. CHAI, M. ISPIR, R. ANIL, Z. HAQUE, L. HONG, V. JAIN, X. LIU, AND H. SHAH, *Wide & deep learning for recommender systems*, in Proc. 1st Workshop on Deep Learning for Recommender Systems, DLRS 2016, 2016, pp. 7–10.

[37] J. CHENG, K. HUANG, AND Z. ZHENG, *Towards better forecasting by fusing near and distant future visions*, in AAAI, 2020, pp. 3593–3600.

[38] X. CHENG, D. MIAO, C. WANG, AND L. CAO, *Coupled term-term relation analysis for document clustering*, in IJCNN'2013, 2013, pp. 1–8.

[39] C. CHUANG, *Application of hybrid case-based reasoning for enhanced performance in bankruptcy prediction*, Inf. Sci., 236 (2013), pp. 174–185.

[40] Q. CUI, S. WU, Q. LIU, W. ZHONG, AND L. WANG, *Mv-rnn: A multi-view recurrent neural network for sequential recommendation*, IEEE Trans. Knowl. Data Eng., (2018), pp. 1–1.

[41] K. DALLEAU, M. COUCEIRO, AND M. SMAÏL-TABBONE, *Unsupervised extra trees: a stochastic approach to compute similarities in heterogeneous data*, Int. J. Data Sci. Anal., 9 (2020), pp. 447–459.

[42] Z. DANG, C. DENG, X. YANG, AND H. HUANG, *Multi-scale fusion subspace clustering using similarity constraint*, in CVPR, 2020, pp. 6657–6666.

[43] S. DASGUPTA, C. F. STEVENS, AND S. NAVLAKHA, *A neural algorithm for a fundamental computing problem*, Science, 358 (2017), pp. 793–796.

[44] M. DATAR, N. IMMORLICA, P. INDYK, AND V. S. MIRROKNI, *Locality-sensitive hashing scheme based on p-stable distributions*, in Symposium on Computational Geometry, ACM, 2004, pp. 253–262.

[45] H. A. DAU, A. J. BAGNALL, K. KAMGAR, C. M. YEH, Y. ZHU, S. GHARGHABI, C. A. RATANAMAHATANA, AND E. J. KEOGH, *The UCR time series archive*, IEEE CAA J. Autom. Sinica, 6 (2019), pp. 1293–1305.

[46] R. L. DE MÁNTARAS, D. MCSHERRY, D. G. BRIDGE, D. B. LEAKE, B. SMYTH, S. CRAW, B. FALTINGS, M. L. MAHER, M. T. COX, K. D. FORBUS, M. T. KEANE, A. AAMODT, AND I. D. WATSON, *Retrieval, reuse, revision and retention in case-based reasoning*, Knowl. Eng. Rev., 20 (2005), pp. 215–240.

[47] A. P. DEMPSTER, N. M. LAIRD, AND D. B. RUBIN, *Maximum likelihood from incomplete data via the em algorithm*, Journal of the Royal Statistical Society, 39 (1977), pp. 1–38.

[48] J. DEVLIN, M. CHANG, K. LEE, AND K. TOUTANOVA, *BERT: pre-training of deep bidirectional transformers for language understanding*, in NAACL-HLT (1), 2019, pp. 4171–4186.

[49] M. D. DILMI, L. BARTHES, C. MALLET, AND A. CHAZOTTES, *Iterative multiscale dynamic time warping (ims-dtw): a tool for rainfall time series comparison*, Int. J. Data Sci. Anal., 10 (2020), pp. 65–79.

[50] K. G. DIZAJI, F. ZHENG, N. SADOUGHI, Y. YANG, C. DENG, AND H. HUANG, *Unsupervised deep generative adversarial hashing network*, in CVPR, 2018, pp. 3664–3673.

[51] T. D. T. DO AND L. CAO, *Metadata-dependent infinite poisson factorization for efficiently modelling sparse and large matrices in recommendation*, in Proc. Twenty-Seventh Int. Joint Conf. Artif. Intell., 2018, pp. 5010–5016.

[52] K. ECHIHABI, *High-dimensional vector similarity search: From time series to deep network embeddings*, in SIGMOD, 2020, pp. 2829–2832.

[53] M. ESPADOTO, R. M. MARTINS, A. KERREN, N. S. T. HIRATA, AND A. C. TELEA, *Toward a quantitative survey of dimension reduction techniques*, IEEE Trans. Vis. Comput. Graph., 27 (2021), pp. 2153–2173.

[54] C. FAN, Y. ZHANG, Y. PAN, X. LI, C. ZHANG, R. YUAN, D. WU, W. WANG, J. PEI, AND H. HUANG, *Multi-horizon time series forecasting with temporal attention learning*, in KDD, ACM, 2019, pp. 2527–2535.

[55] V. FLUNKERT, D. SALINAS, AND J. GASTHAUS, *Deepar: Probabilistic forecasting with autoregressive recurrent networks*, CoRR, abs/1704.04110 (2017).

[56] M. FRIGO AND S. G. JOHNSON, *The design and implementation of FFTW3*, Proc. IEEE, 93 (2005), pp. 216–231.

[57] N. GARCIA-PEDRAJAS, C. HERVAS-MARTINEZ, AND D. ORTIZ-BOYER, *Cooperative coevolution of artificial neural network ensembles for pattern classification*, IEEE Trans. Evol. Comput., 9 (2005), pp. 271–302.

[58] V. GATTUPALLI, Y. ZHUO, AND B. LI, *Weakly supervised deep image hashing through tag embeddings*, in CVPR, 2019, pp. 10375–10384.

[59] J. GEHRING, M. AULI, D. GRANGIER, D. YARATS, AND Y. N. DAUPHIN, *Convolutional sequence to sequence learning*, in Pro. 34th Int. Conf. Machine Learning, 2017, pp. 1243–1252.

[60] K. GEORGIEV AND P. NAKOV, *A non-iid framework for collaborative filtering with restricted boltzmann machines*, in ICML (3), vol. 28 of JMLR Workshop and Conference Proceedings, JMLR.org, 2013, pp. 1148–1156.

[61] Y. GONG, S. LAZEBNIK, A. GORDO, AND F. PERRONNIN, *Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2013), pp. 2916–2929.

[62] Y. GONG AND Q. ZHANG, *Hashtag recommendation using attention-based convolutional neural network*, in Proc. Twenty-Fifth Int. Joint Conf. Artif. Intell., 2016, pp. 2782–2788.

[63] K. GREFF, S. VAN STEENKISTE, AND J. SCHMIDHUBER, *Neural expectation maximization*, in NIPS, 2017, pp. 6691–6701.

[64] D. GU, C. LIANG, I. BICHINDARITZ, C. ZUO, AND J. WANG, *A case-based knowledge system for safety evaluation decision making of thermal power plants*, Knowl.-Based Syst., 26 (2012), pp. 185–195.

[65] D. GU, C. LIANG, AND H. ZHAO, *A case-based reasoning system based on weighted heterogeneous value distance metric for breast cancer diagnosis*, Artif. Intell. Medicine, 77 (2017), pp. 31–47.

[66] C. GUO, H. JIA, AND N. ZHANG, *Time series clustering based on ica for stock data analysis*, in WiMob, 2008, pp. 1–4.

[67] H. GUO, R. TANG, Y. YE, Z. LI, AND X. HE, *Deepfm: A factorization-machine based neural network for CTR prediction*, in Proc. Twenty-Sixth Int. Joint Conf. Artif. Intell., 2017, pp. 1725–1731.

[68] Y. GUO, J. HU, AND Y. PENG, *Research of new strategies for improving CBR system*, Artif. Intell. Rev., 42 (2014), pp. 1–20.

[69] Y. GUO, J. HU, AND Y. PENG, *Research of new strategies for improving CBR system*, Artif. Intell. Rev., 42 (2014), pp. 1–20.

[70] B. HAO, W. W. SUN, Y. LIU, AND G. CHENG, *Simultaneous clustering and estimation of heterogeneous graphical models*, J. Mach. Learn. Res., 18 (2017), pp. 217:1–217:58.

[71] G. HE, H. WANG, S. LIU, AND B. ZHANG, *CSMVC: A multiview method for multivariate time-series clustering*, IEEE Trans. Cybern., (2021), pp. 1–13.

[72] H. HE, Q. ZHANG, S. BAI, K. YI, AND Z. NIU, *CATN: cross attentive tree-aware network for multivariate time series forecasting*, in AAAI, 2022, pp. 4030–4038.

[73] J. HE, S. CHANG, R. RADHAKRISHNAN, AND C. BAUER, *Compact hashing with joint optimization of search accuracy and time*, in CVPR, 2011, pp. 753–760.

[74] R. HE AND J. MCAULEY, *Fusing similarity models with markov chains for sparse sequential recommendation*, in Proc. IEEE 16th Int. Conf. Data Mining, 2016, pp. 191–200.

[75] X. HE AND T. CHUA, *Neural factorization machines for sparse predictive analytics*, in Proc. 40th Int. ACM SIGIR Conf. Res. Develop. in Inf. Retrieval, 2017, pp. 355–364.

[76] X. HE, L. LIAO, H. ZHANG, L. NIE, X. HU, AND T. CHUA, *Neural collaborative filtering*, in WWW, ACM, 2017, pp. 173–182.

[77] J. M. HERNÁNDEZ-LOBATO, N. HOULSBY, Z. GHAHRAMANI, AND , *Probabilistic matrix factorization with non-random missing data*, in ICML, 2014, pp. 1512–1520.

[78] G. E. HINTON, O. VINYALS, AND J. DEAN, *Distilling the knowledge in a neural network*, CoRR, abs/1503.02531 (2015).

[79] H. HONG, Y. LIN, X. YANG, Z. LI, K. FU, Z. WANG, X. QIE, AND J. YE, *Heteta: Heterogeneous information network embedding for estimating time of arrival*, in KDD, 2020, pp. 2444–2454.

[80] K.-H. HSU, *A case-based classifier for hypertension detection*, Knowl.-Based Syst., 24 (2011), pp. 33 – 39.

[81] L. HU, L. CAO, J. CAO, Z. GU, G. XU, AND D. YANG, *Learning informative priors from heterogeneous domains to improve recommendation in cold-start user domains*, ACM Trans. Inf. Syst., 35 (2016), pp. 13:1–13:37.

[82] L. HU, S. JIAN, L. CAO, Z. GU, Q. CHEN, AND A. AMIRBEKYAN, *HERS: modeling influential contexts with heterogeneous relations for sparse and cold-start recommendation*, in AAAI, 2019, pp. 3830–3837.

[83] J. HUANG, K. LUO, L. CAO, Y. WEN, AND S. ZHONG, *Learning multiaspect traffic couplings by multirelational graph attention networks for traffic prediction*, IEEE Trans. Intell. Transp. Syst., 23 (2022), pp. 20681–20695.

[84] J. HUANG, Z. REN, W. X. ZHAO, G. HE, J. WEN, AND D. DONG, *Taxonomy-aware multi-hop reasoning networks for sequential recommendation*, in Proc. Twelfth ACM Int. Conf. Web Search and Data Mining, 2019, pp. 573–581.

[85] Y. HUANG, L. CAO, J. ZHANG, L. PAN, AND Y. LIU, *Exploring feature coupling and model coupling for image source identification*, IEEE Trans. Inf. Forensics Secur., 13 (2018), pp. 3108–3121.

[86] F. ILHAN, O. KARAAHMETOGLU, I. BALABAN, AND S. S. KOZAT, *Markovian rnn: An adaptive time series prediction network with hmm-based switching for nonstationary environments*, IEEE Trans. Neural Networks Learn. Syst., (2021), pp. 1–14.

[87] P. INDYK AND R. MOTWANI, *Approximate nearest neighbors: Towards removing the curse of dimensionality*, in STOC, 1998, pp. 604–613.

[88] V. JALALI AND D. LEAKE, *Harnessing hundreds of millions of cases: Case-based prediction at industrial scale*, in ICCBR, vol. 11156, 2018, pp. 153–169.

[89] V. JALALI AND D. B. LEAKE, *CBR meets big data: A case study of large-scale adaptation rule generation*, in ICCBR, vol. 9343, 2015, pp. 181–196.

[90] S. JIAN, G. PANG, L. CAO, K. LU, AND H. GAO, *CURE: flexible categorical data representation by hierarchical coupling learning*, IEEE Trans. Knowl. Data Eng., 31 (2019), pp. 853–866.

[91] M. JIANG, S. ZHANG, J. HUANG, L. YANG, AND D. N. METAXAS, *Scalable histopathological image analysis via supervised hashing with multiple features*, Medical Image Anal., 34 (2016), pp. 3–12.

[92] Q. JIANG, X. CUI, AND W. LI, *Deep discrete supervised hashing*, IEEE Trans. Image Process., 27 (2018), pp. 5996–6009.

[93] Q. JIANG AND W. LI, *Asymmetric deep supervised hashing*, in AAAI, 2018, pp. 3342–3349.

[94] T. JOACHIMS, A. SWAMINATHAN, AND T. SCHNABEL, *Unbiased learning-to-rank with biased feedback*, in WSDM, 2017, pp. 781–789.

[95] W. KANG, C. FANG, Z. WANG, AND J. J. MCAULEY, *Visually-aware fashion recommendation and design with generative image models*, in ICDM, IEEE Computer Society, 2017, pp. 207–216.

[96] W. KANG AND J. J. MCAULEY, *Self-attentive sequential recommendation*, in ICDM, IEEE Computer Society, 2018, pp. 197–206.

[97] A. KARPATHY, G. TODERICI, S. SHETTY, T. LEUNG, R. SUKTHANKAR, AND F. LI, *Large-scale video classification with convolutional neural networks*, in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.

[98] Y. KATZNELSON, *An introduction to harmonic analysis*, Cambridge University Press„ (1970).

[99] M. R. KHOSRAVANI AND S. NASIRI, *Injection molding manufacturing process: review of case-based reasoning applications*, J. Intell. Manuf., 31 (2020), pp. 847–864.

[100] M. R. KHOSRAVANI, S. NASIRI, AND K. WEINBERG, *Application of case-based reasoning in a fault detection system on production of drippers*, Appl. Soft Comput., 75 (2019), pp. 227–232.

[101] D. H. KIM, C. PARK, J. OH, S. LEE, AND H. YU, *Convolutional matrix factorization for document context-aware recommendation*, in Proc. 10th ACM Conf. Recommender Systems, 2016, pp. 233–240.

[102] Y. KIM, *Convolutional neural networks for sentence classification*, in Proc. 2014 Conf. Empirical Methods Natural Language Processing, 2014, pp. 1746–1751.

[103] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, in ICLR (Poster), 2015.

[104] N. KITAEV, L. KAISER, AND A. LEVSKAYA, *Reformer: The efficient transformer*, in ICLR, 2020.

[105] L. H. KOOPMANS, *The spectral analysis of time series*, Elsevier, 1995.

[106] A. R. KOSIOREK, S. SABOUR, Y. W. TEH, AND G. E. HINTON, *Stacked capsule autoencoders*, in NeurIPS, 2019, pp. 15486–15496.

[107] R. G. KRISHNAN, D. LIANG, AND M. D. HOFFMAN, *On the challenges of learning with inference networks on sparse, high-dimensional data*, in AISTATS, vol. 84, 2018, pp. 143–151.

[108] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep convolutional neural networks*, in NIPS, 2012, pp. 1106–1114.

[109] H. W. KUHN, *The hungarian method for the assignment problem*, Naval research logistics quarterly, 2 (1955), pp. 83–97.

[110] M. J. KUSNER, B. PAIGE, AND J. M. HERNÁNDEZ-LOBATO, *Grammar variational autoencoder*, in ICML, vol. 70 of Proceedings of Machine Learning Research, 2017, pp. 1945–1954.

[111] G. LAI, W. CHANG, Y. YANG, AND H. LIU, *Modeling long- and short-term temporal patterns with deep neural networks*, in SIGIR, 2018, pp. 95–104.

[112] J. LAMY, B. D. SEKAR, G. GUÉZENNEC, J. BOUAUD, AND B. SÉROUSSI, *Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach*, Artif. Intell. Medicine, 94 (2019), pp. 42–53.

[113] H. LANGE, S. L. BRUNTON, AND J. N. KUTZ, *From fourier to koopman: Spectral methods for long-term time series prediction*, J. Mach. Learn. Res., 22 (2021), pp. 41:1–41:38.

[114] S. I. LAO, K. L. CHOY, G. T. S. HO, R. C. M. YAM, Y. C. TSIM, AND T. C. POON, *Achieving quality assurance functionality in the food industry using a hybrid case-based reasoning and fuzzy logic approach*, Expert Syst. Appl., 39 (2012), pp. 5251–5261.

[115] J. LI, H. IZAKIAN, W. PEDRYCZ, AND I. JAMAL, *Clustering-based anomaly detection in multivariate time series data*, Appl. Soft Comput., 100 (2021), p. 106919.

[116] J. LI, P. REN, Z. CHEN, Z. REN, T. LIAN, AND J. MA, *Neural attentive session-based recommendation*, in CIKM, ACM, 2017, pp. 1419–1428.

[117] Q. LI, Z. SUN, R. HE, AND T. TAN, *Deep supervised discrete hashing*, in NIPS, 2017, pp. 2482–2491.

[118] S. LI, X. JIN, Y. XUAN, X. ZHOU, W. CHEN, Y. WANG, AND X. YAN, *Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting*, in NeurIPS, 2019, pp. 5244–5254.

[119] S. LI, X. JIN, Y. XUAN, X. ZHOU, W. CHEN, Y.-X. WANG, AND X. YAN, *Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting*, Advances in neural information processing systems, 32 (2019).

[120] Y. LI, R. YU, C. SHAHABI, AND Y. LIU, *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting*, in ICLR (Poster), 2018.

[121] J. LIAN, X. ZHOU, F. ZHANG, Z. CHEN, X. XIE, AND G. SUN, *xdeepfm: Combining explicit and implicit feature interactions for recommender systems*, in KDD, ACM, 2018, pp. 1754–1763.

[122] D. LIANG, L. CHARLIN, J. MCINERNEY, AND D. M. BLEI, *Modeling user exposure in recommendation*, in WWW, 2016, pp. 951–961.

[123] J. LIM, M.-J. CHAE, Y. YANG, I.-B. PARK, J. LEE, AND J. PARK, *Fast scheduling of semiconductor manufacturing facilities using case-based reasoning*, IEEE Trans. Semiconduct M., 29 (2015), pp. 22–32.

[124] K. LIN, J. XU, I. M. BAYTAS, S. JI, AND J. ZHOU, *Multi-task feature interaction learning*, in KDD, ACM, 2016, pp. 1735–1744.

[125] G. LING, H. YANG, M. R. LYU, AND I. KING, *Response aware model-based collaborative filtering*, in UAI, 2012, pp. 501–510.

[126] R. J. A. LITTLE AND D. B. RUBIN, *Statistical Analysis with Missing Data*, John Wiley & Sons, Inc., New York, NY, USA, 1986.

[127] B. LIU, Y. CAO, M. LONG, J. WANG, AND J. WANG, *Deep triplet quantization*, in ACM Multimedia, 2018, pp. 755–763.

[128] C. LIU AND H. CHEN, *A novel CBR system for numeric prediction*, Inf. Sci., 185 (2012), pp. 178–190.

[129] C.-H. LIU AND H.-C. CHEN, *A novel CBR system for numeric prediction*, Inform. Sciences, 185 (2012), pp. 178 – 190.

[130] H. LIU, R. WANG, S. SHAN, AND X. CHEN, *Deep supervised hashing for fast image retrieval*, Int. J. Comput. Vis., 127 (2019), pp. 1217–1234.

[131] K. LIU, A. BELLET, AND F. SHA, *Similarity learning for high-dimensional sparse data*, in AISTATS, vol. 38, 2015.

[132] S. LIU, H. YU, C. LIAO, J. LI, W. LIN, A. X. LIU, AND S. DUSTDAR, *Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting*, in ICLR, OpenReview.net, 2022.

[133] W. LIU, C. MU, S. KUMAR, AND S. CHANG, *Discrete graph hashing*, in NIPS, 2014, pp. 3419–3427.

[134] W. LIU, J. WANG, R. JI, Y. JIANG, AND S. CHANG, *Supervised hashing with kernels*, in CVPR, 2012, pp. 2074–2081.

[135] W. LIU, J. WANG, S. KUMAR, AND S. CHANG, *Hashing with graphs*, in ICML, 2011, pp. 1–8.

[136] Y. LIU, Y. XIAO, Q. WU, C. MIAO, J. ZHANG, B. ZHAO, AND H. TANG, *Diversified interactive recommendation with implicit feedback*, in AAAI, 2020.

[137] Y. LIU, P. ZHAO, X. LIU, M. WU, L. DUAN, AND X.-L. LI, *Learning user dependencies for recommendation*, in IJCAI, 2017, pp. 2379–2385.

[138] Z. LIU, G. QIU, G. MERCIER, AND Q. PAN, *A transfer classification method for heterogeneous data based on evidence theory*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, (2019), pp. 1–13.

[139] LONGBING CAO, *Coupling learning of complex interactions*, Inf. Process. Manage., 51 (2015), pp. 167–186.

[140] ——, *Data Science Thinking: The Next Scientific, Technological and Economic Revolution*, Data Analytics, Springer International Publishing, 2018.

[141] R. LOPEZ DE MANTARAS, *Retrieval, reuse, revision and retention in case-based reasoning*, Knowl. Eng. Rev., 20 (2005), pp. 215–240.

[142] X. LUO, C. CHEN, H. ZHONG, H. ZHANG, M. DENG, J. HUANG, AND X. HUA, *A survey on deep hashing methods*, CoRR, abs/2003.03369 (2020).

[143] Q. MA, C. CHEN, S. LI, AND G. W. COTTRELL, *Learning representations for incomplete time series clustering*, in AAAI, 2021, pp. 8837–8846.

[144] Q. MA, S. LI, W. ZHUANG, S. LI, J. WANG, AND D. ZENG, *Self-supervised time series clustering with model-based dynamics*, IEEE Trans. Neural Networks Learn. Syst., 32 (2021), pp. 3942–3955.

[145] Q. MA, J. ZHENG, S. LI, AND G. W. COTTRELL, *Learning representations for time series clustering*, in NeurIPS, 2019, pp. 3776–3786.

[146] B. M. MARLIN AND R. S. ZEMEL, *Collaborative prediction and ranking with non-random missing data*, in ACM RecSys, 2009, pp. 5–12.

[147] B. M. MARLIN, R. S. ZEMEL, S. ROWEIS, AND M. SLANEY, *Collaborative filtering and the missing at random assumption*, in UAI, 2007, pp. 267–275.

[148] F. MARTÍNEZ-ÁLVAREZ, A. T. LORA, J. C. RIQUELME, AND J. S. AGUILAR-RUIZ, *Energy time series forecasting based on pattern sequence similarity*, IEEE Trans. Knowl. Data Eng., 23 (2011), pp. 1230–1243.

[149] M. MELUCCI, *Impact of query sample selection bias on information retrieval system ranking*, in DSAA'2016, 2016, pp. 341–350.

[150] N. MISHRA, S. PETROVIC, AND S. SUNDAR, *A self-adaptive case-based reasoning system for dose planning in prostate cancer radiotherapy*, Med. Phys., 38 (2011), pp. 6528–6538.

[151] T. MIYATO, T. KATAOKA, M. KOYAMA, AND Y. YOSHIDA, *Spectral normalization for generative adversarial networks*, in ICLR, 2018.

[152] S. MONTANI, *How to use contextual knowledge in medical case-based reasoning systems: A survey on very recent trends*, Artif. Intell. Medicine, 51 (2011), pp. 125–131.

[153] J. L. MOORE, S. CHEN, D. TURNBULL, AND T. JOACHIMS, *Taste over time: The temporal dynamics of user preferences*, in Proc. 14th Int. Soc. Music Information Retrieval Conf., 2013, pp. 401–406.

[154] J. MUANGPRATHUB, V. BOONJING, AND P. PATTARAINTAKORN, *A new case-based classification using incremental concept lattice knowledge*, Data Knowl. Eng., 83 (2013), pp. 39–53.

[155] N. NGUYEN AND B. QUANZ, *Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting*, in AAAI, AAAI Press, 2021, pp. 9117–9125.

[156] P. NGUYEN, J. WANG, M. HILARIO, AND A. KALOUSIS, *Learning heterogeneous similarity measures for hybrid-recommendations in meta-mining*, in ICDM, 2012, pp. 1026–1031.

[157] S. OHSAWA, Y. OBARA, AND T. OSOGAMI, *Gated probabilistic matrix factorization: Learning users' attention from missing values*, in IJCAI, 2016, pp. 1888–1894.

[158] B. N. ORESHKIN, D. CARPOV, N. CHAPADOS, AND Y. BENGIO, *N-BEATS: neural basis expansion analysis for interpretable time series forecasting*, in ICLR, 2020.

[159] D. PANDOVE, S. GOEL, AND R. RANI, *Systematic review of clustering high-dimensional and large datasets*, ACM Trans. Knowl. Discov. Data, 12 (2018), pp. 16:1–16:68.

[160] G. PANG AND L. CAO, *Heterogeneous univariate outlier ensembles in multidimensional data*, ACM Trans. Knowl. Discov. Data, 14 (2020), pp. 68:1–68:27.

[161] G. PANG, L. CAO, AND L. CHEN, *Outlier detection in complex categorical data by modeling the feature value couplings*, in Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, 2016, pp. 1902–1908.

[162] G. PANG, L. CAO, L. CHEN, AND H. LIU, *Learning representations of ultrahigh-dimensional data for random distance-based outlier detection*, in KDD, 2018, pp. 2041–2050.

[163] J. PAPARRIZOS AND L. GRAVANO, *k-shape: Efficient and accurate clustering of time series*, in SIGMOD, 2015, pp. 1855–1870.

[164] M.-A. PARSEVAL, *Mémoire sur les séries et sur l,Äôintégration complète d,Äôune équation aux différences partielles linéaires du second ordre, à coefficients constants*, Mém. prés. par divers savants, Acad. des Sciences, Paris,(1), 1 (1806), pp. 638–648.

[165] S. PETROVIC, N. MISHRA, AND S. SUNDAR, *A novel case based reasoning approach to radiotherapy planning*, Expert Syst. Appl., 38 (2011), pp. 10759–10769.

[166] W. QIANG, S. YI, AND Q. Z. JIAN, *A nonlinear correlation measure for multivariable data set*, Physica D Nonlinear Phenomena, 200 (2005), pp. 287–295.

[167] Y. QIN, D. SONG, H. CHEN, W. CHENG, G. JIANG, AND G. W. COTTRELL, *A dual-stage attention-based recurrent neural network for time series prediction*, in IJCAI, 2017, pp. 2627–2633.

[168] R. QIU, Z. HUANG, J. LI, AND H. YIN, *Exploiting cross-session information for session-based recommendation with graph neural networks*, ACM Trans. Inf. Syst., 38 (2020), pp. 22:1–22:23.

[169] L. RAKKAPPAN AND V. RAJAN, *Context-aware sequential recommendations with-stacked recurrent neural networks*, in WWW, ACM, 2019, pp. 3172–3178.

[170] S. S. RANGAPURAM, M. W. SEEGER, J. GASTHAUS, L. STELLA, Y. WANG, AND T. JANUSCHOWSKI, *Deep state space models for time series forecasting*, in NeurIPS, 2018, pp. 7796–7805.

[171] S. RENDLE, *Factorization machines*, in Proc. 10th IEEE Int. Conf. Data Mining, 2010, pp. 995–1000.

[172] S. RENDLE, C. FREUDENTHALER, AND L. SCHMIDT-THIEME, *Factorizing personalized markov chains for next-basket recommendation*, in Proc. 19th Int. Conf. WWW, 2010, pp. 811–820.

[173] M. T. REZVAN, A. Z. HAMADANI, AND A. SHALBAFZADEH, *Case-based reasoning for classification in the mixed data sets employing the compound distance methods*, Eng. Appl. Artif. Intell., 26 (2013), pp. 2001–2009.

[174] M. T. REZVAN, A. ZEINAL HAMADANI, AND A. SHALBAFZADEH, *Case-based reasoning for classification in the mixed data sets employing the compound distance methods*, Eng. Appl. Artif. Intell., 26 (2013), pp. 2001–2009.

[175] RUINING HE AND JULIAN MCAULEY, *Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering*, in Proc. 25th Int. Conf. WWW, 2016, pp. 507–517.

[176] L. G. B. RUIZ, M. DEL CARMEN PEGALAJAR JIMÉNEZ, R. ARCUCCI, AND M. MOLINA-SOLANA, *A time-series clustering methodology for knowledge extraction in energy consumption data*, Expert Syst. Appl., 160 (2020), p. 113731.

[177] Y. SAITO, *Asymmetric tri-training for debiasing missing-not-at-random explicit feedback*, in SIGIR, ACM, 2020, pp. 309–318.

[178] R. SALAKHUTDINOV AND G. E. HINTON, *Semantic hashing*, Int. J. Approx. Reason., 50 (2009), pp. 969–978.

[179] R. SALAKHUTDINOV AND A. MNIH, *Probabilistic matrix factorization*, in NIPS, 2007, pp. 1257–1264.

[180] R. SALAKHUTDINOV, A. MNIH, AND G. E. HINTON, *Restricted boltzmann machines for collaborative filtering*, in ICML, vol. 227 of ACM International Conference Proceeding Series, ACM, 2007, pp. 791–798.

[181] D. SALINAS, M. BOHLKE-SCHNEIDER, L. CALLOT, R. MEDICO, AND J. GASTHAUS, *High-dimensional multivariate forecasting with low-rank gaussian copula processes*, in NeurIPS, 2019, pp. 6824–6834.

[182] A. A. SÁNCHEZ-RUIZ AND S. ONTAÑÓN, *Least common subsumer trees for plan retrieval*, in Proc. ICCBR, vol. 8765, 2014, pp. 405–419.

[183] SÁNCHEZ-RUIZ, ANTONIO A. AND ONTAÑÓN, SANTIAGO, *Least common subsumer trees for plan retrieval*, in Proc. ICCBR, Cork, Ireland, 2014, pp. 405–419.

[184] F. SARTORI, A. MAZZUCCHELLI, AND A. D. GREGORIO, *Bankruptcy forecasting using case-based reasoning: The creperie approach*, Expert Syst. Appl., 64 (2016), pp. 400–411.

[185] T. SCHNABEL, A. SWAMINATHAN, A. SINGH, N. CHANDAK, AND T. JOACHIMS, *Recommendations as treatments: Debiasing learning and evaluation*, in ICML, 2016, pp. 1670–1679.

[186] A. SCHWABE, J. PERSSON, AND S. FEUERRIEGEL, *Predicting COVID-19 spread from large-scale mobility data*, in KDD, ACM, 2021, pp. 3531–3539.

[187] M. W. SEEGER, D. SALINAS, AND V. FLUNKERT, *Bayesian intermittent demand forecasting for large inventories*, in NIPS, 2016, pp. 4646–4654.

[188] R. SEN, H. YU, AND I. S. DHILLON, *Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting*, in NeurIPS, 2019, pp. 4838–4847.

[189] M. SHARMA, *Design of brain-computer interface-based classification model for mining mental state of covid-19 afflicted mariner's.*, VM Media SP. zo.o VM Group SK, (2020).

[190] F. SHEN, C. SHEN, W. LIU, AND H. T. SHEN, *Supervised discrete hashing*, in CVPR, 2015, pp. 37–45.

[191] S. SHI, W. MA, M. ZHANG, Y. ZHANG, X. YU, H. SHAN, Y. LIU, AND S. MA, *Beyond user embedding matrix: Learning to hash for modeling large-scale users in recommendation*, in SIGIR, 2020, pp. 319–328.

[192] X. SHI, F. XING, K. XU, M. SAPKOTA, AND L. YANG, *Asymmetric discrete graph hashing*, in AAAI, 2017, pp. 2541–2547.

[193] S. C. K. SHIU AND S. K. PAL, *Case-based reasoning: Concepts, features and soft computing*, Appl. Intell., 21 (2004), pp. 233–238.

[194] B. SMYTH, M. T. KEANE, AND P. CUNNINGHAM, *Hierarchical case-based reasoning integrating case-based and decompositional problem-solving techniques for plant-control software design*, IEEE Trans. Knowl. Data Eng., 13 (2001), pp. 793–812.

[195] B. SMYTH, M. T. KEANE, AND P. CUNNINGHAM, *Hierarchical case-based reasoning integrating case-based and decompositional problem-solving techniques for plant-control software design*, IEEE Trans. Knowl. Data Eng., 13 (2001), pp. 793–812.

[196] H. SONG, D. RAJAN, J. J. THIAGARAJAN, AND A. SPANIAS, *Attend and diagnose: Clinical time series analysis using attention models*, in AAAI, 2018, pp. 4091–4098.

[197] J. SONG, Y. YANG, Y. YANG, Z. HUANG, AND H. T. SHEN, *Inter-media hashing for large-scale retrieval from heterogeneous data sources*, in SIGMOD, 2013, pp. 785–796.

[198] G. SPADON, S. HONG, B. BRANDOLI, S. MATWIN, J. F. RODRIGUES-JR, AND J. SUN, *Pay attention to evolution: Time series forecasting with deep graph-evolution learning*, IEEE Trans. Pattern Anal. Mach. Intell., (2021), pp. 1–1.

[199] N. SREBRO, J. D. M. RENNIE, AND T. S. JAAKKOLA, *Maximum-margin matrix factorization*, in NIPS, 2004, pp. 1329–1336.

[200] H. STECK, *Item popularity and recommendation accuracy*, in ACM RecSys, 2011, pp. 125–132.

[201] S. SU, C. ZHANG, K. HAN, AND Y. TIAN, *Greedy hash: Towards fast optimization for accurate hash coding in CNN*, in NeurIPS, 2018, pp. 806–815.

[202] A. SWAMINATHAN AND T. JOACHIMS, *The self-normalized estimator for counterfactual learning*, in NIPS, 2015, pp. 3231–3239.

[203] T. D. T. DO AND L. CAO, *Gamma-poisson dynamic matrix factorization embedded with metadata influence*, in Proc. Thirty-second: Ann. Conf. Neural Information Processing Systems, 2018, pp. 5829–5840.

[204] F. TANG, M. FAN, AND P. TIÑO, *Generalized learning riemannian space quantization: A case study on riemannian manifold of SPD matrices*, IEEE Trans. Neural Networks Learn. Syst., 32 (2021), pp. 281–292.

[205] J. TANG AND K. WANG, *Personalized top-n sequential recommendation via convolutional sequence embedding*, in Proc. Eleventh ACM Int. Conf. Web Search Data Mining, 2018, pp. 565–573.

[206] Y. TAY, A. T. LUU, AND S. C. HUI, *Multi-pointer co-attention networks for recommendation*, in Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, 2018, pp. 2309–2318.

[207] P. S. THOMAS AND E. BRUNSKILL, *Data-efficient off-policy policy evaluation for reinforcement learning*, in ICML, 2016, pp. 2139–2148.

[208] D. T. TRAN, A. IOSIFIDIS, J. KANNIAINEN, AND M. GABBOUJ, *Temporal attention-augmented bilinear network for financial time-series data analysis*, IEEE Trans. Neural Networks Learn. Syst., 30 (2019), pp. 1407–1418.

[209] I. TSAMARDINOS, G. BORBOUDAKIS, P. KATSOGRIDAKIS, P. PRATIKAKIS, AND V. CHRISTOPHIDES, *A greedy feature selection algorithm for big data of high dimensionality*, Mach. Learn., 108 (2019), pp. 149–202.

[210] A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. KAISER, AND I. POLOSUKHIN, *Attention is all you need*, in NIPS, 2017, pp. 5998–6008.

[211] A. VIRMAUX AND K. SCAMAN, *Lipschitz regularity of deep neural networks: analysis and efficient estimation*, in NeurIPS, 2018, pp. 3839–3848.

[212] DIEDERIK P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, in ICLR (Poster), 2015.

[213] HARALD STECK, *Evaluation of recommendations: rating-prediction and ranking*, in ACM RecSys, 2013, pp. 213–220.

[214] JOHN PAPARRIZOS AND L. GRAVANO, *Fast and accurate time-series clustering*, ACM Trans. Database Syst., 42 (2017), pp. 8:1–8:49.

[215] STEFFEN RENDLE, *Factorization machines with libfm*, ACM TIST, 3 (2012), pp. 57:1–57:22.

[216] B. WANG, J. LU, Z. YAN, H. LUO, T. LI, Y. ZHENG, AND G. ZHANG, *Deep uncertainty quantification: A machine learning approach for weather forecasting*, in KDD, ACM, 2019, pp. 2087–2095.

[217] C. WANG, Z. SHE, AND L. CAO, *Coupled attribute analysis on numerical data*, in IJCAI'2013, 2013, pp. 1736–1742.

[218] H. WANG, E. SKAU, H. KRIM, AND G. CERVONE, *Fusing heterogeneous data: A case for remote sensing and social media*, IEEE Trans. Geosci. Remote. Sens., 56 (2018), pp. 6956–6968.

[219] J. WANG AND J. CAVERLEE, *Recurrent recommendation with local coherence*, in WSDM, ACM, 2019, pp. 564–572.

[220] J. WANG, S. KUMAR, AND S. CHANG, *Semi-supervised hashing for large-scale search*, IEEE Trans. Pattern Anal. Mach. Intell., 34 (2012), pp. 2393–2406.

[221] J. WANG, W. LIU, S. KUMAR, AND S. CHANG, *Learning to hash for indexing big data - A survey*, Proceedings of the IEEE, 104 (2016), pp. 34–57.

[222] J. WANG, T. ZHANG, J. SONG, N. SEBE, AND H. T. SHEN, *A survey on learning to hash*, IEEE Trans. Pattern Anal. Mach. Intell., 40 (2018), pp. 769–790.

[223] L. WANG, Y. WU, AND R. LI, *Quantile regression for analyzing heterogeneity in ultra-high dimension*, J. AM. STAT. ASSOC., 107 (2012), pp. 214–222.

[224] M. WANG, M. GONG, X. ZHENG, AND K. ZHANG, *Modeling dynamic missingness of implicit feedback for recommendation*, in NIPS, 2018, pp. 6670–6679.

[225] M. WANG, X. ZHENG, Y. YANG, AND K. ZHANG, *Collaborative filtering with social exposure: A modular approach to social recommendation*, in AAAI, 2018, pp. 2516–2523.

[226] P. WANG, Y. FAN, L. XIA, W. X. ZHAO, S. NIU, AND J. HUANG, *KERL: A knowledge-guided reinforcement learning model for sequential recommendation*, in SIGIR, ACM, 2020, pp. 209–218.

[227] R. WANG, B. FU, G. FU, AND M. WANG, *Deep & cross network for ad click predictions*, in ADKDD@KDD, ACM, 2017, pp. 12:1–12:7.

[228] R. WANG, R. WANG, S. QIAO, S. SHAN, AND X. CHEN, *Deep position-aware hashing for semantic continuous image retrieval*, in WACV, 2020, pp. 2482–2491.

[229] S. WANG, L. HU, L. CAO, X. HUANG, D. LIAN, AND W. LIU, *Attention-based transactional context embedding for next-item recommendation*, in Proc. Thirty-Second AAAI Conf. Artif. Intell., 2018.

[230] S. WANG, L. HU, Y. WANG, L. CAO, Q. Z. SHENG, AND M. A. ORGUN, *Sequential recommender systems: Challenges, progress and prospects*, in IJCAI, ijcai.org, 2019, pp. 6332–6338.

[231] W. WANG, Y. XU, J. SHEN, AND S. ZHU, *Attentive fashion grammar network for fashion landmark detection and clothing category classification*, in Proc. IEEE Conf. Computer Vision Pattern Recognition, 2018, pp. 4271–4280.

[232] X. WANG, X. HE, M. WANG, F. FENG, AND T. CHUA, *Neural graph collaborative filtering*, in SIGIR, ACM, 2019, pp. 165–174.

[233] X. WANG, R. ZHANG, Y. SUN, AND J. QI, *Doubly robust joint learning for recommendation on data missing not at random*, in ICML, 2019, pp. 6638–6647.

[234] X. WANG, Z. ZHANG, B. WU, F. SHEN, AND G. LU, *Prototype-supervised adversarial network for targeted attack of deep hashing*, in CVPR, 2021, pp. 16357–16366.

[235] Y. WANG, A. SMOLA, D. C. MADDIX, J. GASTHAUS, D. FOSTER, AND T. JANUSCHOWSKI, *Deep factors for forecasting*, in ICML, vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 6607–6617.

[236] R. WARLOP, A. LAZARIC, AND J. MARY, *Fighting boredom in recommender systems with linear reinforcement learning*, in Proc. NIPS, 2018, pp. 1764–1773.

[237] M. W. WATSON, *Vector autoregressions and cointegration*, Working Paper Series, Macroeconomic Issues, 4 (1993).

[238] Y. WEISS, A. TORRALBA, AND R. FERGUS, *Spectral hashing*, in NIPS, 2008, pp. 1753–1760.

[239] J. WOODBRIDGE, B. MORTAZAVI, A. A. T. BUI, AND M. SARRAFZADEH, *Improving biomedical signal search results in big data case-based reasoning environments*, Pervasive Mob. Comput., 28 (2016), pp. 69–80.

[240] C. WU, A. AHMED, A. BEUTEL, A. J. SMOLA, AND H. JING, *Recurrent recommender networks*, in Proc. Tenth ACM Int. Conf. Web Search and Data Mining, 2017, pp. 495–503.

[241] C. F. J. WU, *On the convergence properties of the em algorithm*, The Annals of Statistics, 11 (1983), pp. 95–103.

[242] D. WU, Q. DAI, J. LIU, B. LI, AND W. WANG, *Deep incremental hashing network for efficient image retrieval*, in CVPR, 2019, pp. 9069–9077.

[243] H. WU, J. XU, J. WANG, AND M. LONG, *Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting*, in NeurIPS, 2021, pp. 22419–22430.

[244] S. WU, Y. TANG, Y. ZHU, L. WANG, X. XIE, AND T. TAN, *Session-based recommendation with graph neural networks*, in AAAI, AAAI Press, 2019, pp. 346–353.

[245] Y. WU, S. C. H. HOI, T. MEI, AND N. YU, *Large-scale online feature selection for ultra-high dimensional sparse data*, ACM Trans. Knowl. Discov. Data, 11 (2017), pp. 48:1–48:22.

[246] Z. WU, S. PAN, F. CHEN, G. LONG, C. ZHANG, AND P. S. YU, *A comprehensive survey on graph neural networks*, IEEE Trans. Neural Networks Learn. Syst., 32 (2021), pp. 4–24.

[247] Z. WU, S. PAN, G. LONG, J. JIANG, X. CHANG, AND C. ZHANG, *Connecting the dots: Multivariate time series forecasting with graph neural networks*, in KDD, 2020, pp. 753–763.

[248] Z. WU, S. PAN, G. LONG, J. JIANG, AND C. ZHANG, *Graph wavenet for deep spatial-temporal graph modeling*, in IJCAI, 2019, pp. 1907–1913.

[249] XIANGNAN HE AND TAT-SENG CHUA, *Neural factorization machines for sparse predictive analytics*, in SIGIR, 2017, pp. 355–364.

[250] J. XIAO, H. YE, X. HE, H. ZHANG, F. WU, AND T. CHUA, *Attentional factorization machines: Learning the weight of feature interactions via attention networks*, in Proc. 26th Int. J. Conf. Artif. Intell. IJCAI, 2017, pp. 3119–3125.

[251] C. XU, P. ZHAO, Y. LIU, J. XU, V. S. SHENG, Z. CUI, X. ZHOU, AND H. XIONG, *Recurrent convolutional neural network for sequential recommendation*, in WWW, ACM, 2019, pp. 3398–3404.

[252] J. XU AND L. CAO, *High-dimensional cross-market dependence modeling and portfolio forecasting by copula variational LSTM*, SSRN, (2021), pp. 1–44.

[253] J. XU, W. WEI, AND L. CAO, *Copula-based high dimensional cross-market dependence modeling*, in DSAA, 2017, pp. 734–743.

[254] H. XUE, X. DAI, J. ZHANG, S. HUANG, AND J. CHEN, *Deep matrix factorization models for recommender systems*, in IJCAI, ijcai.org, 2017, pp. 3203–3209.

[255] A. YAN, H. SHAO, AND Z. GUO, *Weight optimization for case-based reasoning using membrane computing*, Inform. Sciences, 287 (2014), pp. 109 – 120.

[256] A. YAN, W. WANG, C. ZHANG, AND H. ZHAO, *A fault prediction method that uses improved case-based reasoning to continuously predict the status of a shaft furnace*, Inf. Sci., 259 (2014), pp. 269–281.

[257] H. YANG, K. LIN, AND C. CHEN, *Supervised learning of semantics-preserving hash via deep convolutional neural networks*, IEEE Trans. Pattern Anal. Mach. Intell., 40 (2018), pp. 437–451.

[258] H. YANG, G. LING, Y. SU, M. R. LYU, AND I. KING, *Boosting response aware model-based collaborative filtering*, IEEE Trans. Knowl. Data Eng., 27 (2015), pp. 2064–2077.

[259] J. YANG AND J. LESKOVEC, *Patterns of temporal variation in online media*, in WSDM, 2011, pp. 177–186.

[260] L. YANG, Y. CUI, Y. XUAN, C. WANG, S. J. BELONGIE, AND D. ESTRIN, *Unbiased offline recommender evaluation for missing-not-at-random implicit feedback*, in ACM RecSys, 2018, pp. 279–287.

[261] T. YANG, R. YAO, Q. YIN, AND O. WU, *Mitigating sentimental bias via a polar attention mechanism*, Int J Data Sci Anal, (2020).

[262] W. YE, S. WANG, X. CHEN, X. WANG, Z. QIN, AND D. YIN, *Time matters: Sequential recommendation with complex temporal information*, in SIGIR, ACM, 2020, pp. 1459–1468.

[263] YEHUDA KOREN, *Collaborative filtering with temporal dynamics*, in Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2009, pp. 447–456.

[264] B. YU, H. YIN, AND Z. ZHU, *Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting*, in IJCAI, 2018, pp. 3634–3640.

[265] H. YU, C. HSIEH, S. SI, AND I. S. DHILLON, *Scalable coordinate descent approaches to parallel matrix factorization for recommender systems*, in Proc. 12th IEEE Int. Conf. Data Mining, 2012, pp. 765–774.

[266] H. YU, N. RAO, AND I. S. DHILLON, *Temporal regularized matrix factorization for high-dimensional time series prediction*, in NIPS, 2016, pp. 847–855.

[267] F. YUAN, A. KARATZOGLOU, I. ARAPAKIS, J. M. JOSE, AND X. HE, *A simple convolutional generative network for next item recommendation*, in WSDM, ACM, 2019, pp. 582–590.

[268] L. YUAN, T. WANG, X. ZHANG, F. E. H. TAY, Z. JIE, W. LIU, AND J. FENG, *Central similarity quantization for efficient image and video retrieval*, in CVPR, 2020, pp. 3080–3089.

[269] W. YUAN, H. WANG, X. YU, N. LIU, AND Z. LI, *Attention-based context-aware sequential recommendation model*, Inf. Sci., 510 (2020), pp. 122–134.

[270] J. ZAKARIA, A. MUEEN, AND E. J. KEOGH, *Clustering time series using unsupervised-shapelets*, in ICDM, 2012, pp. 785–794.

[271] G. ZERVEAS, S. JAYARAMAN, D. PATEL, A. BHAMIDIPATY, AND C. EICKHOFF, *A transformer-based framework for multivariate time series representation learning*, in KDD, 2021, pp. 2114–2124.

[272] H. ZHA, X. HE, C. H. Q. DING, M. GU, AND H. D. SIMON, *Spectral relaxation for k-means clustering*, in NIPS, 2001, pp. 1057–1064.

[273] K. ZHANG, B. HUANG, J. ZHANG, C. GLYMOUR, AND B. SCHÖLKOPF, *Causal discovery from nonstationary/heterogeneous data: Skeleton estimation and orientation determination*, in IJCAI, 2017, pp. 1347–1353.

[274] L. ZHANG, C. C. AGGARWAL, AND G. QI, *Stock price prediction via discovering multi-frequency trading patterns*, in KDD, 2017, pp. 2141–2149.

[275] Q. ZHANG, L. CAO, C. SHI, AND Z. NIU, *Neural time-aware sequential recommendation by jointly modeling preference dynamics and explicit feature couplings*, IEEE Trans. Neural Networks Learn. Syst., (2021), pp. 1–13.

[276] Q. ZHANG, L. CAO, C. ZHU, Z. LI, AND J. SUN, *Coupledcf: Learning explicit and implicit user-item couplings in recommendation for deep collaborative filtering*, in Proc. 23rd Int. Joint Conf. Artif. Intell., 2018, pp. 3662–3668.

[277] Q. ZHANG, C. SHI, Z. NIU, AND L. CAO, *HCBC: A hierarchical case-based classifier integrated with conceptual clustering*, IEEE Trans. Knowl. Data Eng., 31 (2019), pp. 152–165.

[278] Q. ZHANG, C. SHI, P. SUN, AND Z. NIU, *Case-based classification on hierarchical structure of formal concept analysis*, in ECAI, vol. 285, 2016, pp. 1758–1759.

[279] S. ZHANG, L. YAO, A. SUN, AND Y. TAY, *Deep learning based recommender system: A survey and new perspectives*, ACM Comput. Surv., 52 (2019), pp. 5:1–5:38.

[280] Z. ZHANG, L. LIAO, M. HUANG, X. ZHU, AND T. CHUA, *Neural multimodal belief tracker with adaptive attention for dialogue systems*, in WWW, ACM, 2019, pp. 2401–2412.

[281] G. ZHAO, Y. XIONG, L. CAO, D. LUO, X. SU, AND Y. ZHU, *A cost-effective lsh filter for fast pairwise mining*, in ICDM'2009, 2009, pp. 1088–1093.

[282] Q. ZHAO, P. N. BENNETT, A. FOURNEY, A. L. THOMPSON, S. WILLIAMS, A. D. TROY, AND S. T. DUMAIS, *Calendar-aware proactive email recommendation*, in Proc. 41st Int ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2018, pp. 655–664.

[283] S. ZHAO, T. ZHAO, H. YANG, M. R. LYU, AND I. KING, *STELLAR: spatial-temporal latent ranking for successive point-of-interest recommendation*, in Proc. Thirtieth AAAI Conf. Artif. Intell., 2016, pp. 315–322.

[284] B. ZHENG, X. ZHAO, L. WENG, N. Q. V. HUNG, H. LIU, AND C. S. JENSEN, *PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search*, Proc. VLDB Endow., 13 (2020), pp. 643–655.

[285] J. ZHENG, J. LIU, C. SHI, F. ZHUANG, J. LI, AND B. WU, *Recommendation in heterogeneous information network via dual similarity regularization*, Int. J. Data Sci. Anal., 3 (2017), pp. 35–48.

[286] Y. ZHENG, X. YI, M. LI, R. LI, Z. SHAN, E. CHANG, AND T. LI, *Forecasting fine-grained air quality based on big data*, in KDD, 2015, pp. 2267–2276.

[287] W. ZHONG, S. ROHATGI, J. WU, C. L. GILES, AND R. ZANIBBI, *Accelerating substructure similarity search for formula retrieval*, in ECIR (1), vol. 12035, 2020, pp. 714–727.

[288] G. ZHOU, W. BIAN, K. WU, L. REN, Q. PI, Y. ZHANG, C. XIAO, X. SHENG, N. MOU, X. LUO, C. ZHANG, X. QIAO, S. XIANG, K. GAI, X. ZHU, AND J. XU, *CAN: revisiting feature co-action for click-through rate prediction*, CoRR, abs/2011.05625 (2020).

[289] H. ZHOU, S. ZHANG, J. PENG, S. ZHANG, J. LI, H. XIONG, AND W. ZHANG, *Informer: Beyond efficient transformer for long sequence time-series forecasting*, in AAAI, 2021, pp. 11106–11115.

[290] T. ZHOU, Z. MA, Q. WEN, X. WANG, L. SUN, AND R. JIN, *Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting*, in ICML, vol. 162 of Proceedings of Machine Learning Research, PMLR, 2022, pp. 27268–27286.

[291] X. ZHOU, C. MASCOLO, AND Z. ZHAO, *Topic-enhanced memory networks for personalised point-of-interest recommendation*, in KDD, ACM, 2019, pp. 3018–3028.

[292] C. ZHU, L. CAO, Q. LIU, J. YIN, AND V. KUMAR, *Heterogeneous metric learning of categorical data with hierarchical couplings*, IEEE Transactions on Knowledge and Data Engineering, 30 (2018), pp. 1254–1267.

[293] C. ZHU, L. CAO, AND J. YIN, *Unsupervised heterogeneous coupling learning for categorical representation*, IEEE Trans. Pattern Anal. Mach. Intell., 44 (2022), pp. 533–549.

[294] C. ZHU, Q. ZHANG, L. CAO, AND A. ABRAHAMYAN, *Mix2vec: Unsupervised mixed data representation*, in 7th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2020, Sydney, Australia, October 6-9, 2020, IEEE, 2020, pp. 118–127.

[295] G. ZHU, J. HU, J. QI, J. MA, AND Y. PENG, *An integrated feature selection and cluster analysis techniques for case-based reasoning*, Eng. Appl. Artif. Intel., 39 (2015), pp. 14–22.

[296] G.-N. ZHU, J. HU, J. QI, J. MA, AND Y.-H. PENG, *An integrated feature selection and cluster analysis techniques for case-based reasoning*, Eng. Appl. Artif. Intell., 39 (2015), pp. 14 – 22.

[297] H. ZHU, M. LONG, J. WANG, AND Y. CAO, *Deep hashing network for efficient similarity retrieval*, in AAAI, 2016, pp. 2415–2421.

[298] L. ZHU, X. LU, Z. CHENG, J. LI, AND H. ZHANG, *Flexible multi-modal hashing for scalable multimedia retrieval*, ACM Trans. Intell. Syst. Technol., 11 (2020), pp. 14:1–14:20.

[299] P. ZHU, H. CHENG, Q. HU, Q. WANG, AND C. ZHANG, *Towards generalized and efficient metric learning on riemannian manifold*, in IJCAI, 2018, pp. 3235–3241.

[300] Q. ZHU, X. ZHOU, Z. SONG, J. TAN, AND L. GUO, *DAN: deep attention neural network for news recommendation*, in AAAI, AAAI Press, 2019, pp. 5973–5980.

[301] X. ZHU, X. LI, S. ZHANG, Z. XU, L. YU, AND C. WANG, *Graph PCA hashing for similarity search*, IEEE Trans. Multim., 19 (2017), pp. 2033–2044.

[302] Y. ZHU, X. ZHANG, R. WANG, W. ZHENG, AND Y. ZHU, *Self-representation and PCA embedding for unsupervised feature selection*, World Wide Web, 21 (2018), pp. 1675–1688.