Expert Systems With Applications 296 (2026) 128929

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Shrinking for success: Multimodal dimensionality reduction for sustainable recommender systems

Angelo Geninatti Cossatin, Noemi Mauro *

University of Torino, Corso Svizzera, 185, Torino, 10149, Italy

ARTICLE INFO

Keywords:
Multimodal recommender systems
Transformer
Attention bottlenecks
Sustainable computing

ABSTRACT

Multimodal item representations are widely used in modern recommender systems to capture various aspects of items. However, the high dimensionality of these representations poses challenges in terms of computational efficiency and resource usage. In this paper, we propose a fusion method, named MDR, based on attention bottlenecks to obtain condensed multimodal item representations. Our approach, intersecting dimensionality reduction and information fusion, leverages a Transformer autoencoder architecture to learn a compact item representation that preserves the salient information from the original high-dimensional features. We evaluate the impact of the condensed representations on the recommendation performance and resource consumption of a set of recommendation algorithms using the Cornac framework. Experiments on four datasets show that the condensed item representations produced by our fusion method enable the recommender algorithms to achieve comparable or even improved recommendation performance compared to the original, high-dimensional representations. Moreover, using the condensed representations significantly reduces training time, RAM Memory usage, and GPU utilization. These findings highlight the potential of our approach to enhance the efficiency and sustainability of multimodal recommender systems without compromising their effectiveness.

1. Introduction

Recommender systems employ multiple information sources about users and items to generate personalized suggestions. Many modern recommendation algorithms leverage multimodal item representations that capture aspects like textual descriptions, images, audio, and more. However, representing items through high-dimensional multimodal features introduces challenges concerning computational efficiency and resource usage that raise the need for an efficient fusion of multimodal data while keeping the dimensionality manageable. Moreover, by reducing the dimensionality of item representations, we can save resources and reduce energy consumption and the emissions produced while training the models. This is in line with the Sustainable Development Goals (SDGs) defined in 2015 in the 2030 Agenda for Sustainable Development.[1]

Dimensionality reduction techniques are crucial for addressing the challenges posed by high-dimensional multimodal data in recommender systems. While various methods exist, such as SVD or autoencoders, they often struggle to effectively capture the complex relationships across different modalities. Our work explores the intersection of dimensionality reduction and information fusion, proposing a novel approach, named MDR, that combines multimodal information while significantly reducing its dimensionality. More specifically, we propose a fusion method to obtain condensed multimodal item representations using a Transformer architecture which leverages Attention Bottlenecks (Nagrani et al., 2021). These bottlenecks serve as controlled communication channels between different modalities during training, helping overcome the limitations of traditional dimensionality reduction methods mentioned above. The goal is to reduce the dimensionality of the multimodal feature vectors while still preserving the salient information. Transformer architectures are well-suited for this task due to their ability to process variable-length inputs, which enables efficient handling of diverse multimodal features. We aim to reduce the computational cost of the recommendation process, making it more energy-efficient, and, thus, more environmentally sustainable. We developed MDRto be agnostic to the recommendation algorithm, producing condensed representations that may be used regardless of the recommendation pipeline. Moreover, MDRis trained via unsupervised learning, therefore it does not require annotated datasets with

* Corresponding author.
E-mail address: noemi.mauro@unito.it (N. Mauro).

[1] https://sdgs.un.org/2030agenda

https://doi.org/10.1016/j.eswa.2025.128929
Received 20 September 2024; Received in revised form 16 June 2025; Accepted 3 July 2025
Available online 6 July 2025
0957-4174/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

condensed representations. To make this possible, we train the model by asking it to reconstruct the original input starting from the condensed representation.

The main research questions we aim to address are:

1. Do the condensed item representations obtained through the proposed fusion method (MDR) preserve or improve recommendation performance compared to the original high-dimensional versions?
2. Can the condensed representations obtained with MDR help save computational resources in terms of RAM and GPU memory usage, and training and test times?

To evaluate MDR, we conducted experiments using the Cornac framework[2] (Truong et al., 2021a) with different types of recommender system algorithms. We assessed their performance when using the original high-dimensional item representations, compared to the condensed ones obtained by applying our method. We used metrics like rating prediction error, ranking performance, and resource consumption for this evaluation.

Our results show that MDR significantly reduces the dimensionality of the multimodal item representations while largely preserving and, in some cases, improving the performance of the recommendation algorithms. Moreover, the condensed representations lead to substantial savings in computational resources required by these algorithms.

To summarize the contributions of this paper are:

- Development of a novel fusion-based dimensionality reduction model (MDR) for condensing high-dimensional multimodal item representations in recommender systems.
- Evaluating recommendation performance and resource usage of different recommendation algorithms that incorporate a condensed representation of multimodal information obtained with MDR. As a baseline, we use the original high-dimensional item representation to understand if our approach saves computational resources by preserving the recommendation performance of the algorithms.

The remainder of this paper is organized as follows. Section 2 discusses related work, covering background on information fusion and dimensionality reduction methods and their application in recommender systems. Section 3 presents our proposed model, detailing its architecture. Section 7 describes the experimental setup to evaluate the proposed approach. Section 4 provides information about the four datasets used. Sections 8 and 9 present and analyze the results obtained from the experiment. Section 10 discusses the limitations and future work and Section 11 concludes the paper.

The code is available at https://github.com/angelogeninatti/shrinking_for_success_code/.

2. Related work

2.1. Dimensionality reduction

Dimensionality reduction techniques play a crucial role in addressing the challenges posed by high-dimensional multimodal data in recommender systems. These methods aim to reduce the complexity of data while preserving its essential characteristics, thereby improving computational efficiency and mitigating the curse of dimensionality. In the context of recommendation systems, dimensionality reduction is particularly valuable for handling large-scale, multimodal item representations. Below are some of the most widely used SoA dimensionality reduction techniques.

Singular value decomposition (SVD) is a linear algebra technique widely used in recommender systems for both dimensionality reduction and collaborative filtering (Ba et al., 2013; Guan et al., 2017; Reddy & Adilakshmi, 2014; Zarzour et al., 2018; Zhou et al., 2015). SVD decomposes a matrix into three matrices, allowing for the extraction of latent factors that represent users and items in a lower-dimensional space (Klema & Laub, 1980). This approach has been particularly effective in addressing the sparsity problem in collaborative filtering recommendation systems (Guan et al., 2017; Zarzour et al., 2018).

UMAP is a non-linear dimensionality reduction technique. It preserves both local and global structure of the data, making it particularly suitable for capturing complex relationships in high-dimensional spaces (Mcinnes et al., 2018). In the context of recommendation, UMAP has demonstrated effectiveness in reducing the dimensionality of user-item interaction data while maintaining the system's performance. For instance, research has shown that UMAP with just two dimensions can achieve results comparable to other methods using 25 dimensions in tracking model tendencies (Li & Zhan, 2023).

Autoencoders are a type of neural network architecture which can learn non-linear mappings for dimensionality reduction. They consist of an encoder that compresses the input data into a lower-dimensional representation and a decoder that reconstructs the original input from this compressed representation. Autoencoders have been widely used in the context of recommendation for multiple tasks, including feature extraction and dimensionality reduction (Zhang et al., 2020).

Matryoshka representation learning (MRL) is a dimensionality reduction technique that encodes information at multiple levels of detail within a single embedding. MRL creates hierarchical representations where smaller embeddings are contained within larger ones, with each level capturing increasingly fine-grained information (Kusupati et al., 2022).

Deep hashing is another widely used technique which can be leveraged to obtain compact representations with the goal of optimizing storage. It is versatile and lightweight, making it adaptable to any context, both supervised and unsupervised (Luo et al., 2023).

However, traditional dimensionality reduction techniques often fail to adequately account for different modalities when dealing with multimodal data. In contrast, our proposed method looks at each modality separately, accounting for their complex interactions through controlled communication channels when generating a condensed representation. Unlike approaches that merge modalities early in the process (like SVD and Autoencoders) or create general-purpose nested representations (like MRL and deep hashing methods), our method specifically preserves the distinct characteristics of each modality while still capturing their meaningful relationships through attention bottlenecks.

2.2. Information fusion

Multimodal information fusion employs various techniques to combine features from different modalities, aiming to generate a unified representation that enhances the performance of machine learning models.

Researchers have explored different approaches for fusing multimodal data in recommender systems. Some studies employ basic concatenation or addition for early (Oramas et al., 2017b; Wang et al., 2021; Wei et al., 2019a) or late fusion (Bracher et al., 2016; Malitesta et al., 2024; Yuan et al., 2023), sometimes using a Multi-Layer Perceptron to refine the outcomes of these operations (Tian et al., 2021), while others utilize more sophisticated approaches. Torres (2023) introduces a late fusion method that leverages user preferences, similar to the objectives of other studies (Wang et al., 2023a). Deldjoo et al. propose extending the Borda count rank aggregation strategy to perform late fusion on independently extracted modalities (Deldjoo et al., 2018). Han et al. present an intermediate fusion method employing an encoder-decoder neural architecture (Han et al., 2022).

The application of attention mechanisms to perform fusion for recommender systems has been explored in multiple works (Chen et al., 2019; Liu et al., 2023; Zhou et al., 2020). However, to our knowledge, none have investigated the use of attention bottlenecks in

[2] Cornac is one of the suggested frameworks by the ACM Conference on Recommender Systems: https://github.com/ACMRecSys/recsys-evaluation-frameworks

dimensionality reduction for multimodal recommendation. Attention bottlenecks involve allowing communication between modalities only through a limited number of bottleneck tokens shared across modalities and updated after each encoding step. This allows for learning cross-modal interactions without increasing the computational cost of training as much as allowing the interaction between all dimensions of modalities. Moreover, it has good performance with classification, which is a supervised task (Nagrani et al., 2021).

In this work, we aim to bridge this gap in the literature by proposing a Transformer model that learns a condensed representation for items through unsupervised learning and exploiting attention bottlenecks. Notice that the approach we developed can be used in a recommendation task to obtain a lower-dimensional representation of items without requiring the usage of annotated datasets for model training.

Typically, information fusion mechanisms, including attention-based ones, are exclusively employed to obtain a richer, more exhaustive item representation. Differently, our goal is to obtain a combined representation of items in a lower-dimensional space to reduce the computational cost of machine learning algorithms (notably, recommendation algorithms), without decreasing their performance.

Moreover, MDRis agnostic to the recommendation algorithm and produces condensed representations that may be used regardless of the recommendation pipeline.

2.3. Multimodal recommender systems

Multimodal recommender systems leverage diverse information sources to enrich item representations and better capture user preferences. These systems integrate multiple modalities to understand complex item characteristics that single-modality approaches might miss. Recent advancements in multimodal embedding techniques span several paradigms (Liu et al., 2024). Matrix factorization approaches jointly process multiple modalities to learn shared latent representations (Park et al., 2017). Deep learning architectures, particularly variational autoencoders, have enhanced these capabilities by employing Bayesian generative models that connect user ratings with multimodal content (Li & She, 2017; Liang et al., 2024). Visual-aware recommendation approaches demonstrate the value of incorporating image features into preference modeling frameworks (He & Mcauley, 2016b; Li et al., 2021). Graph-based methods have gained prominence for their ability to capture complex connectivity patterns in user-item interaction graphs enhanced with multimodal features (Guo et al., 2024; Wei et al., 2019b; Zhang et al., 2021; Zhou et al., 2023; Zhou & Shen, 2023), while diffusion models leverage probabilistic processes to effectively propagate information across heterogeneous modalities (Jiang et al., 2024; Wang et al., 2023). Moreover, recent advances in sequential recommendation have leveraged contrastive learning to improve the quality of learned representations (Zhang et al., 2025). Despite these advancements, most approaches still struggle with the computational costs of high-dimensional multimodal representations, highlighting the need for efficient dimension-reduction techniques that preserve cross-modal relationships. Specifically, our research focuses on comparing how existing multimodal recommendation algorithms perform when using either high-dimensional concatenated embeddings or condensed ones obtained with MDR, rather than on developing a new multimodal recommendation algorithm. Thus, we designed our approach to be compatible with various recommendation algorithms.

3. Proposed model: MDR

To obtain condensed representations of multi-modal information, we exploited a fusion method based on the concept of attention bottlenecks within a Transformer autoencoder. The problem can be formalized as follows.

Given $1, \dots, N$ modalities (extracted from, e.g., textual, visual, or audio information) of a multi-modal high-dimensional representation of the item I, the embeddings $M1, \dots, MN$ of the item I are represented as follows:

$$I = [M1 = [m1_0, m1_1, \dots, m1_k], \dots, MN = [mn_0, mn_1, \dots, mn_l]]$$

In this work, we aim to build a condensed representation C of I,

$$C = [c_0, \dots, c_j] \; where \; j << k + \dots + l$$

i.e., the dimensionality of the condensed representation is significantly lower than the sum of the dimensionalities of all modalities in the original representation of I.

More specifically, the architecture of MDRis composed of:

- The encoding module: a set of Transformer Encoder layers, and a Fusion Bottleneck Tokens Generator, that process each modality of the multi-modal information input separately (Fig. 1).
- The decoding module: a set of Transformer Decoder layers that aims to reconstruct the original input from the condensed representation.

As previously mentioned, we employed an unsupervised technique to guide the model training without relying on annotated datasets. Specifically, we used Smooth L1 Loss to measure the reconstruction error and used the Transformer as an autoencoder.

3.1. Encoding module

The encoding module is built with a stack of encoding layers, one for each modality of the multi-modal information. The input of each stack of encoding layers is an embedding of the data concerning the related modality.[3]

The encoder module leverages the following components to process the input (Fig. 1):

1. A positional encoder injects information about the position of each token in the input sequence into the input embeddings (Section 3.1.1). This is necessary because, as the Transformer architecture processes all tokens in parallel, it does not capture positional information by itself.
2. A fusion bottleneck tokens generator creates a set of shared tokens (one for each modality) to support the communication between modalities (Section 3.1.2). These tokens are concatenated to the input embeddings and iteratively updated after each encoding layer.
3. The standard implementation of a stack of Transformer Encoder Layers, leveraging attention bottlenecks instead of the traditional self-attention mechanism.
4. A linear transformation layer for each modality at the end of the stack of encoder layers. We compute the average of the outputs of the linear layers to obtain the condensed representation of the desired dimension.

We do not detail components 3 and 4 as they are part of the standard implementation of the Transformer architecture.

3.1.1. Positional encoder

Transformers are typically used to process sequential input: the positional encoder injects the information about the position of each dimension into the input embeddings, allowing the model to capture spatial information (Vaswani et al., 2017). In MDR, we consider each dimension of the embedding as a token in the input sequence. This enables the positional encoder to identify and encode the position of each embedding

[3] Each modality is preceded by a classification token (unique for each modality) and a Start of String token. Conversely, each modality is followed by an End of String token. This allows the model to distinguish the modalities.

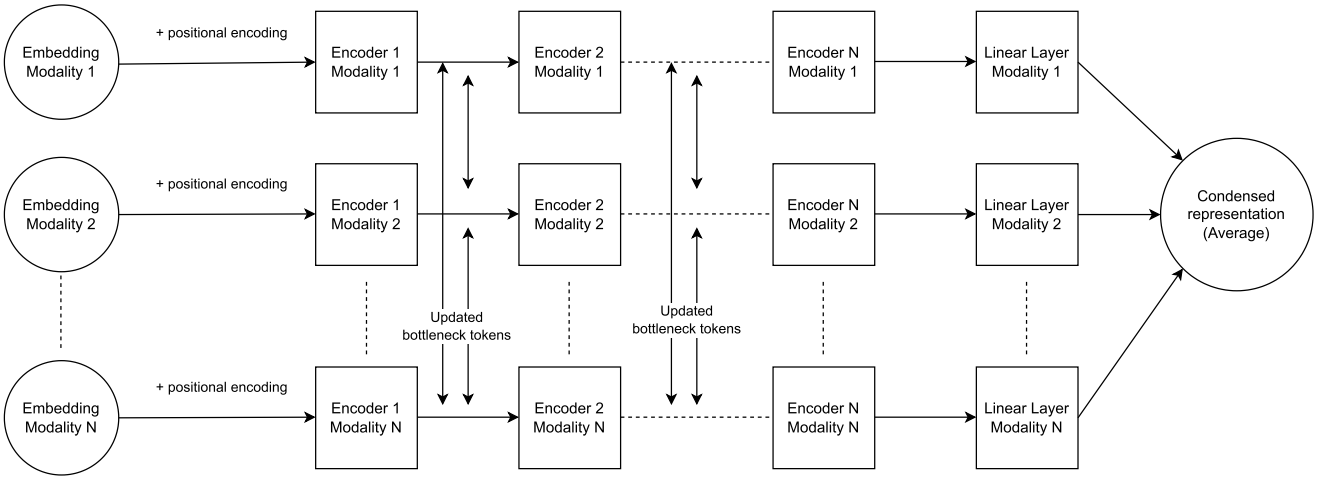


Fig. 1. Diagram representing the encoding phase.

dimension. We chose to use the standard positional encoding formula (Vaswani et al., 2017):

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right)$$

where pos is the position index, i is the dimension index, and d_{model} is the model dimension. To improve training robustness, we applied dropout regularization.

3.1.2. Fusion bottleneck tokens generator (via attention)

The Fusion Bottleneck Tokens Generator enables cross-modal communication within the MDR architecture. This module is orthogonal to the others in the encoding step. Unlike traditional multi-modal approaches that either process modalities independently or concatenate them, the bottleneck mechanism creates a controlled information exchange pathway. Specifically, the generator initializes a set of learnable tokens (one set per modality). During forward propagation, these tokens are appended to each modality's input sequence before entering the encoder stack. As information flows through successive encoder layers, the bottleneck tokens accumulate modality-specific contextual information through self-attention. After each encoder layer l processes a $modality_i$, we extract the updated bottleneck token representation $\hat{z}_{fsn_i}^{(l+1)}$ from the encoder output. These modality-specific bottleneck tokens are then averaged across all modalities to create a shared representation $z_{fsn}^{(l+1)}$, which is used as input for the next encoder layer. This controlled communication channel ensures that modalities can influence each other's representations, but only through the information distilled into the bottleneck tokens. The gradual refinement of these tokens through multiple encoder layers allows the model to discover meaningful cross-modal correlations while maintaining computational efficiency, as the communication overhead scales with the number of bottleneck tokens rather than the full dimensionality of the input representations.

The formula for updating the bottleneck tokens is:

$$\left[\mathbf{z}_{\text{i}}^{l+1} \| \hat{\mathbf{z}}_{\text{fsn}_{\text{i}}}^{l+1}\right] = \text{Transformer}\left(\left[\mathbf{z}_{\text{i}}^{l} || \mathbf{z}_{\text{fsn}}^{l}\right]; \theta_{\text{i}}\right)$$
$$\mathbf{z}_{\text{fsn}}^{l+1} = \text{Avg}_{\text{i}}\left(\hat{\mathbf{z}}_{\text{fsn}_i}^{l+1}\right)$$

Here, l is the current encoding layer, z_i is the representation of instance z for modality i, $\hat{\mathbf{z}}_{\text{fsn}_{\text{i}}}^{l+1}$ are the fusion bottleneck tokens as output of layer l for modality i, $||$ is the concatenation operation, and θ_i are the model parameters for modality i.

3.2. Decoding module

The decoding module consists of a single Transformer Decoder layer that takes the condensed representations as input and aims to reconstruct the original input embeddings. This allows the model to learn to preserve the salient information from the original high-dimensional embeddings in the condensed representation. We use a single decoder for all modalities instead of separate decoders for each modality for two reasons: first, it significantly reduces the computational cost of the proposed architecture; second, by limiting the capabilities of the decoder, we encourage the encoder to find a meaningful condensed representation. The output of the decoder then passes through a linear layer. The end result is then compared with the concatenation of the different input modalities, aiming to reduce the difference between the two.

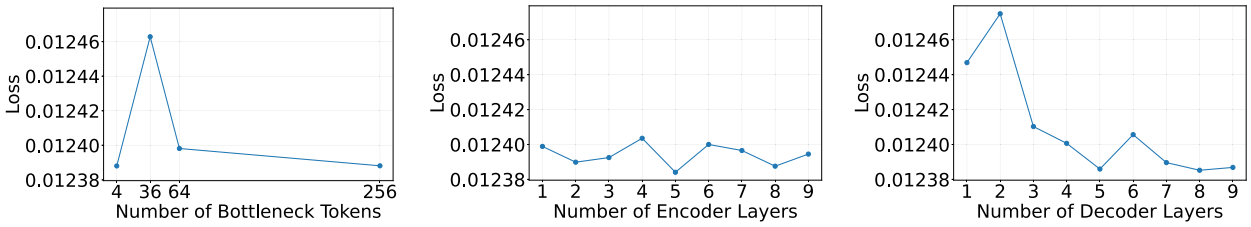
4. Data

We tested MDRon four different datasets, representing heterogeneous domains, which we detail below. Please note that we leveraged precomputed embeddings to enhance reproducibility and to avoid causing further emissions from calculating the embeddings from scratch.

Movielens enhanced. The original Movielens 100K dataset contains 100,000 ratings from 1000 users on 1700 movies (GroupLens Research , 2015). The enhanced version also includes audio, text, and video embeddings for each item. In this work, we exploited the compressed version, containing 500 dimensions for each modality (Pingali, 2023).

Amazon clothing (reduced). We exploited the 2014 version of McAuley's Amazon product data dataset (He & Mcauley, 2016a). It contains metadata about items, reviews, and extracted image features, represented as vectors of 4096 dimensions. We used the 5-core version of the dataset. To extract embeddings from textual data we used Gensim's Doc2Vec model: We obtained vectors of 500 dimensions from each item's title and description (Řehůřek, 2023). To reduce the computational cost of our experiments, we used a subset of the "Clothing, Shoes and Jewelry" category from the 5-core dataset, filtered on the clothes category; this subset includes 5108 items, 30.216 users and 74,266 reviews.

Amazon sports. We also conducted our experiments with the Sports category of the Amazon product data dataset (He & Mcauley, 2016a), containing 18,188 items, 35,598 users and 293,037 reviews (from the 5-core version of the dataset). We chose it to get a better understanding of how our proposed architecture behaves with a more heterogeneous dataset: indeed, its subcategories include "Clothing", "Hunting & Fishing", "Cycling", and "Exercise & Fitness". We used the same modalities as the Amazon clothing dataset, i.e., we leveraged the existing visual embeddings and extracted textual ones via Doc2Vec (see above).



(a) Bottleneck Tokens (b) Encoder Layers (c) Decoder Layers

Fig. 2. Ablation study results for the MuMu dataset showing the effect of varying bottleneck tokens, encoder layers, and decoder layers on model loss.

Enhanced MuMu. We aimed to experiment on a dataset with high embedding dimensionality, high heterogeneity in the size of each modality, and a larger number of items. Therefore, we combined information from multiple datasets on music[4]:

- The original MuMu dataset, which maps song data from the Million Songs Dataset to the respective reviews (via their album) from the Amazon dataset (Oramas et al., 2017a);
- The metadata from the Amazon dataset (He & Mcauley, 2016a);
- An additional dataset including embeddings from albums (Mccallum et al., 2022).

After combining and filtering these sources, we only kept the 5-core version of the dataset, resulting in 23,165 items, 57,259 users and 320,925 reviews. Each item of the dataset is composed by three modalities:

- The first, with 1000 dimensions, contains information about the artist;
- The second, with 1727 dimensions, contains information about the features of the song;
- The third, with 5 dimensions, contains information about the song metadata (year, key, loudness, tempo, and time signature).

5. Experiments with dimensionality reduction techniques

In this first phase, we aimed to compare the performance of MDRto traditional dimensionality reduction techniques, namely SVD, Autoencoders (both standard and with deep hashing), MRL, and a basic Transformer. We excluded UMAP because it is unsuitable for high-dimensional datasets.

First, we tuned the hyperparameters of all models via 100 iterations of Random Search with 5-fold cross-validation. For each split, we used 20 % of the training set for early stopping. We tuned the following hyperparameters for MDR: number of encoder layers (1 to 8), number of decoder layers (1 to 8), dropout probability (0.1 to 0.3), number of bottleneck tokens (4, 36, 64 or 256), dimension of the condensed representation (from 100 to 300 with step 2), initial learning rate (10^{-2} to 10^{-5}). The training phase was conducted using Adam as optimizer (Kingma & Ba, 2017), with Smooth L1 Loss and Early Stopping with patience set to 5 and min_delta to 0.1. Instead, for SVD we tuned the number of components (10 to 100) and the number of iterations (5 to 20), while for both versions of the autoencoder we tuned the encoding dimension (2 to 64), the hidden layer dimension (32 to 256), the learning rate (0.01 to 0.0001), and the number of epochs (10 to 100). Conversely, for MRL we tuned max_encoding_dim (64 to 512), num_levels (2 to 4), learning_rate (10^{-2} to 10^{-4}), batch_size (64, 128, or 256), num_epochs (10 to 100). The basic Transformer was tuned on the same parameter space as MDR, except for the number of bottleneck tokens (since it had none).

In this context, the Smooth L1 Loss, which we used to guide training for all three models, can be interpreted as a reconstruction error of the original, high-dimensional, embeddings starting from the condensed ones. Therefore, we also employed it to assess the quality of the output generated by MDR, compared to standard dimensionality reduction techniques, for the enhanced MuMu dataset, which has heterogeneous modalities, is the biggest, and is the one with the highest number of dimensions.

The best configuration for MDR led to a reconstruction error of 0.012, compared to 0.023 for SVD, 0.028 for a standard autoencoder, 0.020 for MRL, 0.025 for a standard transformer, and 0.071 for an autoencoder with deep hashing. This provides evidence of the efficacy of MDR in the context of multimodality for dimensionality reduction.

6. Experiments with architectural variations of MDR

6.1. Ablation studies

Once we obtained the best configuration for MDR, we performed an additional analysis on the hyperparameters which determine the architecture of MDR. We varied the best configuration for each of those parameters and analyzed the impact of those variations. Namely:

- The number of bottleneck tokens. We used the space defined in Nagrani et al. (2021) (4, 36, 64, and 256).
- The number of encoder layers: from 1 to 9.
- The number of decoder layers: from 1 to 9.

Fig. 2 shows the results for the MuMu dataset, demonstrating negligible oscillations. The results for the other datasets are in Appendix B and show the same trends. This fundamentally corroborates our findings from Section 6.3, demonstrating the robustness of MDR to hyperparameter variation.

6.2. Aggregation and decoding strategy

As discussed in Section 3, in our model, we made specific assumptions on the architecture, namely:

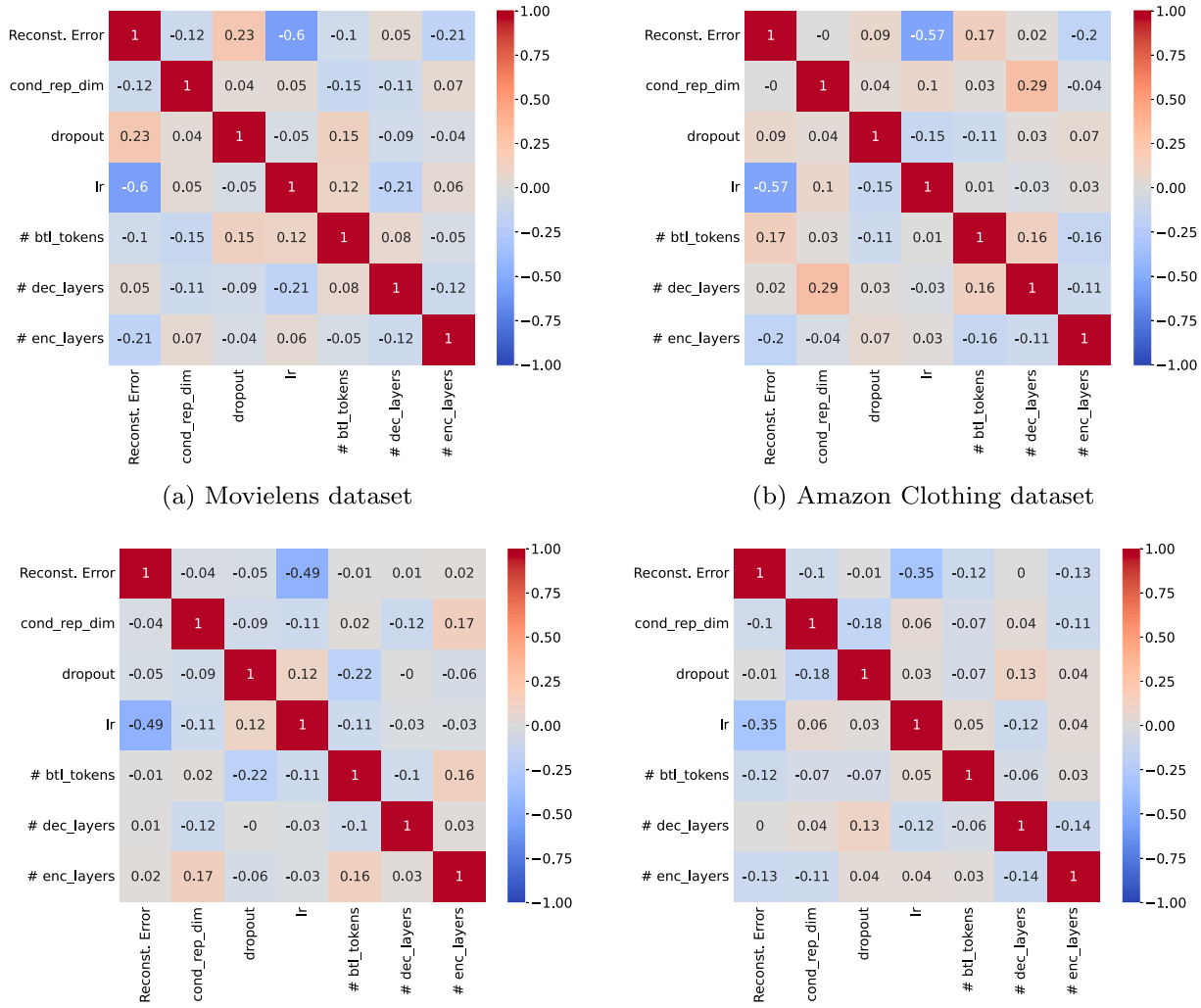
- We averaged the condensed representations for each modality at the end of the encoding step to obtain the final one;
- We used a single set of decoders rather than one for each modality.

We tested these against other possible approaches on all four datasets. More specifically, we experimented with the following aggregation strategies:

- Average. The one we used in MDR.
- Weighted Average, with learned weights during learning.
- Max Pooling, where we used the highest value for each dimension.

We performed random search on the same hyperparameter space as the one we described in Section 5. The random search we performed yielded the results, which are presented in Table 1. The average strategy proved to be the one with the lowest reconstruction error.

[4] The dataset is available upon request



(a) Movielens dataset

(b) Amazon Clothing dataset

(c) Amazon Sports dataset

(d) MuMu dataset

Fig. 3. Correlation heatmaps of loss and hyperparameters for all datasets. To make the image more legible, the names of the hyperparameters have been abbreviated. Reconst. Error is the reconstruction error, cond_rep_dim is condensed_representation_dimension, # enc_layers and # dec_layers are, respectively, num_encoder_layers and num_decoder_layers, and # btl_tokens is num_bottleneck_tokens.

Table 1
Reconstruction errors for different aggregation strategies.

	Movielens	Clothing	Sports	MuMu
Average (MDR)	**0.581648**	**0.285158**	**0.266685**	**0.012374**
Weighted Average	0.581651	0.285252	0.266691	0.012380
Max Pooling	0.581651	0.286697	0.277040	0.012385

Table 2
Reconstruction errors for different decoding strategies.

	Movielens	Clothing	Sports	MuMu
Single	**0.581648**	**0.285158**	**0.266685**	**0.012374**
Separate	0.581649	0.285528	0.266781	0.012391

Regarding the decoding strategy, we compared our approach with a more complex architecture, where each modality has a set of decoders. Results are in Table 2. Again, our architecture choice was confirmed to be solid, obtaining a lower reconstruction error for all datasets. Moreover, it is important to note that having multiple sets of decoders would unnecessarily add complexity to the architecture, resulting in longer training and testing times.

6.3. Analysis of hyperparameter tuning

As a further opportunity to reduce computational resource consumption, we consider possible changes to the execution of Random Search carried out to tune hyperparameters in MDR. We thus investigate ways to focus the search on a much narrower range of values, possibly leading to an optimum hyperparameter configuration in fewer iterations. To assess the impact of varying hyperparameters on the reconstruction error, we plotted the correlation heatmap using Pearson's coefficient (Fig. 3), and the boxplots correlating each hyperaparameter to the reconstruction error (Appendix A). The results indicate that increasing the learning rate has a significant impact on the model's performance; in contrast, the model is robust to variations in the other hyperparameters. More specifically, as shown in Fig. 3 there is a strong negative correlation between the learning rate and the reconstruction error for the Movielens and Amazon Clothing datasets, and a moderate negative correlation for the Amazon Sports and the MuMu ones. This means that increasing the learning rate will lead to a lower reconstruction error. All other correlations (either positive or negative) are very weak, showing the model's robustness to the variation of all hyperparameters except the learning rate. This is also reflected in the boxplots (Appendix A),

Table 3
Best hyperparameters for each dataset.

	Condensed representation dimension	dropout_ p	lr	# bottleneck tokens	# decoder layers	# encoder layers
Movielens	292	0.358	0.01	256	5	6
Amazon Clothing	284	0.195	0.01	256	7	8
Amazon Sports	128	0.397	0.01	64	6	3
MuMu	190	0.384	0.01	36	6	4

where a clear trend indicating higher learning rates lead to better results emerges, while no other clear influences of other parameters can be observed.

As proved by the plots and stated above, the robustness to varying hyperparameters progressively increases with the dataset size, but may already be observed on the smallest Movielens dataset. Moreover, the best parameters are similar across datasets, especially between those having similar sizes (Table 3). These findings suggest that performing random search may not be necessary to achieve good results; the best configuration found during our experiments may serve as a good starting point. If random search is performed, it may be focused on a much narrower range of values, potentially leading to an optimum in fewer iterations. Furthermore, if the dataset used is sufficiently large, the impact of varying hyperparameters diminishes.

7. Experiments with recommendation algorithms

Given the results discussed in Section 5, we tested whether using the condensed representation of the embeddings produced by `MDR` in the context of recommender systems allows to reduce the usage of resources (time, RAM, and GPU memory) without compromising recommendation quality.

7.1. Metrics

For this purpose, we evaluated various recommendation algorithms (see Section 7.3) across four heterogeneous datasets. This evaluation involved comparing their recommendation performance and computational resource consumption using the original, high-dimensional item representation to that using a condensed representation of items generated by our proposed model. Specifically, we were interested in understanding whether:

1. The recommendation performance results obtained by using the condensed representation of the embeddings are similar or better to the ones obtained with the original item representations (research question 1), and
2. The usage of computational resources is lower when the algorithms exploit our condensed item representations (research question 2).

For the evaluation, we considered the following metrics:

- Recommendation performance: MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), Precision@10, Recall@10, NDCG@10 (Normalized Discounted Cumulative Gain), MAP (Mean Average Precision).
- Resources usage: Train time, Test time, Average and Peak RAM utilization, Average and Peak GPU utilization.

7.2. Equivalence test and interpretation of results

To assess whether the condensed item representations and the high-dimensional ones induce similar recommendation performance on the selected algorithms, we conducted a two-one-sided equivalence test using bootstrapping (Belsley et al., 2009). Please, note that different from other statistical tests that are commonly used, the equivalence test does not assess if two (or more) experimental conditions are statistically different, but the contrary; specifically, a **low *p-value* indicates statistical equivalence** between the two conditions, that is, a similar impact on results. We decided to consider the standard threshold for significance, i.e. $p\text{-value} < 0.05$.

We will interpret the results of the experiments by considering three factors: the recommendation performance results, the *p-values* describing statistical equivalence, and resource consumption. Specifically, we will evaluate the results of the experiments as positive when the recommendation performance results obtained by using the condensed representation of the embeddings are similar or better to the ones obtained with the original item representations and the resource consumption is lower. More precisely, we will consider results as positive in the following cases:

1. The *p-value* for a recommendation performance metric is < 0.05 but the resource consumption is lower. In this case, the statistical equivalence test conveys that the recommendation performance via the condensed representation is preserved compared to the results obtained using the high-dimensional one. Moreover, the algorithm uses a lower amount of computational resources. In the tables, this is denoted with †.
2. The recommendation performance for a metric obtained using the condensed representation of the embeddings is higher than that obtained with the original item representations, the *p-value* is ≥ 0.05 and the resource consumption is lower. In the tables, this is denoted with ⋆.

7.3. Algorithms

To compare the performance of the recommendation algorithms with and without condensed item representation, we fed each representation as input to the algorithms. To exploit the condensed item representations, the output from `MDR`(i.e., the condensed embeddings) was directly fed into the algorithms. In contrast, the high-dimensional item representations were concatenated into a single final embedding. We tested our approach on the following algorithms that were embedded in the Cornac framework (Truong et al., 2021a) for the reproducibility of results. The algorithms we chose are highly heterogeneous, ranging from pure content-based to hybrid algorithms, including both neural and non-neural methods.

- Matrix Co-Factorization (MCF), a collaborative filtering approach that jointly factorizes multiple matrices, each representing a different data modality, to learn shared latent representations (Park et al., 2017).
- Collaborative Deep Learning (CDL), a hierarchical Bayesian model that integrates stacked denoising autoencoders (SDAE) into probabilistic matrix factorization (Wang et al., 2015).
- Collaborative Variational Autoencoder (CVAE), a Bayesian generative model considering user ratings and multimodal content (Li & She, 2017).
- Visual Bayesian Personalized Ranking (VBPR), a visually-aware recommendation model that incorporates visual features extracted from product images into the matrix factorization framework (He & Mcauley, 2016b).
- Bilateral Variational Autoencoder for Collaborative Filtering (BiVAECF), a variant of variational autoencoder that learns the joint distribution over user preferences and multimodal item features (Truong et al., 2021b).
- LayerGCN, a graph convolutional network-based recommender system that leverages layer-wise propagation and aggregation to capture high-order connectivity patterns in user-item interaction graphs, enhancing recommendation accuracy and efficiency (Zhou et al., 2023). In our implementation, we used the item representations

(either high-dimensional concatenated embedding or condensed one obtained with MDR) as node features in the graph.
- LGMrec, a multi-graph learning model that captures complex connectivity patterns in heterogeneous user-item graphs by integrating multimodal features with multiple relation types and employing cross-graph message propagation for improved recommendation performance (Guo et al., 2024).
- DiffMM, a diffusion-based multimodal recommender system that leverages probabilistic diffusion processes to effectively propagate and integrate information across heterogeneous modalities, enabling more robust cross-modal preference modeling (Jiang et al., 2024).

We employed 5-fold cross-validation to compare the recommendation performance of the different recommendation algorithms using the condensed and high-dimensional item representations respectively. Please note that, for consistency, the splits for cross-validation were randomly chosen and kept the same across the two experimental conditions. We performed 100 iterations of Random Search on the following hyperparameters, optimizing for the MAE metric:

- MCF: k (1 to 50), max_iter (5 to 100), learning_rate (0.0001 to 0.1).
- CDL: k (1 to 50), autoencoder_structure (100, 200 or 300), max_iter (1 to 100).
- CVAE: z_dim (1 to 100), vae_layers ([200, 100], [300, 200], or [400, 300]), act_fn ('sigmoid', 'relu', or 'tanh'), lr (0.001 to 0.1), n_epochs (5 to 100), lambda_u (0.0001 to 0.01), lambda_v (0.001 to 0.1), lambda_r (1 to 10), lambda_w (0.0001 to 0.01),
- VBPR: k (1 to 50), k2 (1 to 50), n_epochs (5 to 100).
- BiVAECF: k (1 to 50), n_epochs (5 to 100).
- LayerGCN: k (2 to 128), max_iter (5 to 100), learning_rate (0.001 to 0.1), lambda_reg (0.001 to 0.1), n_layers (1 to 10), dropout (0 to 0.5),
- LGMRec: k (1 to 50), feat_embed_dim (1 to 50), n_mm_layers (1 to 5), n_ui_layers (1 to 5), learning_rate (0.001 to 0.1), max_iter (5 to 100), batch_size (128, 256 or 512),
- DiffMM: k (1 to 100), max_iter (5 to 100), learning_rate (0.001 to 0.002), lambda_reg (0.0001 to 0.0002), n_layers (1 to 3), keepRate (0.5 to 0.55), noise_scale (0.01 to 0.02).

8. Results

We report the recommendation performance and the resource usage when running the selected recommendation algorithms in the Cornac framework on the Movielens (Tables 4, 5), Clothing (Tables 6, 7), Sports (Tables 8, 9), and MuMu (Tables 10, 11) datasets. In these tables, each recommendation performance metric, measured for both experimental conditions, is accompanied by the *p-value* of the equivalence test, which, as stated in Subsection 7.2, indicates the likelihood of the results being comparable.

8.1. Impact on recommendation performance

To address the first research question, ("Do the condensed item representations obtained through the proposed fusion method preserve or improve recommendation performance compared to the original high-dimensional versions?"), we compared the recommendation performance of the selected recommendation algorithms using the condensed item representations obtained through MDRagainst the original high-dimensional representations. Across all datasets, the execution of the algorithms using condensed representations achieved comparable or even better recommendation performance than with the original high-dimensional ones in terms of error metrics (MAE and RMSE), relevance metrics (Precision@10, Recall@10), and ranking metrics (MAP and NDCG@10).

For the Movielens dataset (Table 4), the condensed representations outperformed the original ones in most cases, particularly for the VBPR algorithm. Most equivalence *p-values* were low, indicating a high likelihood of equivalence between the two results, with two notable exceptions where equivalence cannot be proved but the results are better for the condensed representation: (i) the error metrics are lower for the results obtained using the condensed representation for VBPR and DiffMM; (ii) the relevance and ranking metrics indicate comparable or superior results when using the condensed embeddings for CDL, MCF, VBPR, CVAE, LightGCN, and LGMRec. We only registered a negative outcome for the relevance and ranking metrics for BiVAECF and LightGCN, where the original high-dimensional embeddings induce superior results (also in this case, results are not equivalent according to the equivalence test). Conversely, for DiffMM, ranking metrics were better when using the condensed embeddings.

Similarly, on the Amazon Clothing dataset (Table 6), the recommendation algorithms achieved better or comparable results when using the condensed representations than the original ones across all metrics and algorithms, with the following exceptions: (i) in terms of ranking metrics, CVAE scored better with the condensed embeddings; (ii) different from what we observed for the Movielens dataset, VBPR and LGMRec obtained better error metrics with the original embeddings.

The results for the Amazon Sports dataset (Table 8) also showed promising recommendation performance for the condensed representations. When using the original representation, VBPR and LGMRec performed better in terms of error metrics. Concerning the other performance metrics, the recommendation algorithms obtained comparable results.

Finally, the findings for the MuMu dataset (Table 10) are consistent with the ones reported for the other datasets. These results indicate a comparable recommendation performance in most cases with notable improvements in the relevance and ranking metrics for the condensed embeddings. Specifically, the condensed embeddings led to lower error metrics for CVAE, higher Precision@10 for MCF and VBPR, and higher Recall@10 for CDL, VBPR, and BiVAECF. In terms of ranking metrics, they led to better results for VBPR, BiVAECF and LGMRec (which also registered better outcomes for relevance metrics), with an improvement in NDCG@10 for MCF and CDL. Conversely, the original embeddings scored better in terms of error metrics for VBPR and LGMRec and relevance and ranking metrics for CVAE; moreover, the Recall@10 and MAP were better for MCF.

In summary, in most cases, we obtained comparable or better recommendation performance when using the condensed item representations. These findings suggest that the condensed item representations generated by MDRcan effectively preserve and frequently enhance the recommendation performance compared to the original high-dimensional representations.

8.2. Impact on resource consumption

To address the second research question ("Can the condensed representations help save computational resources in terms of RAM and GPU memory usage, and training and test times?"), we analyzed the resource consumption of the recommendation algorithms when using the condensed item representations versus the original high-dimensional ones. The resource metrics considered include training time, testing time, average and Peak RAM Memory usage, and average and peak GPU memory usage.

Before describing the results, it is important to note some limitations regarding the reported GPU usage measurements. We employed the GPUtil library (Mortensen, 2024) to measure GPU utilization. However, GPU memory is always allocated in pages; this explains the repeated values appearing in the measurements reported in the tables.

These factors contribute to the occasional reporting of 0 standard deviations in the GPU usage measurements and the possibility that the reported values are higher than the actual GPU utilization.

Table 4
Results on the Movielens dataset. The columns indicate the results of the recommender systems when using the condensed representations (Cond.) and the original (Orig.) ones. Specifically, † means that the *p-value* < 0.05, thus the results using the condensed and original representations are comparable. Moreover, ⋆ means that the *p-value* ≥ 0.05 but the measurement for the condensed representation is better than that of the high-dimensional one. In the round parentheses, we report the standard deviation.

Model	Error						Relevance					
	MAE			RMSE			Precision@10			Recall@10		
	Cond.	Orig.	p-value	Cond.	Orig.	p-value	Cond.	Orig.	p-value	Cond.	Orig.	p-value
MCF	0.753 (0.0047)	**0.753** (0.0058)	0.0†	**0.916** (0.007)	0.919 (0.0064)	0.0†	**0.0428** (0.0314)	0.0391 (0.0261)	0.339⋆	**0.0195** (0.0137)	0.0176 (0.0112)	0.153⋆
CDL	0.947 (0.0122)	**0.94** (0.0175)	0.0†	1.13 (0.0146)	**1.12** (0.0201)	0.0†	0.173 (0.0032)	**0.177** (0.0078)	0.06	0.122 (0.0031)	**0.125** (0.0064)	0.002†
CVAE	0.897 (0.0074)	**0.897** (0.0187)	0.0†	1.09 (0.015)	**1.09** (0.0209)	0.0†	**0.19** (0.0067)	0.187 (0.009)	0.087⋆	**0.126** (0.0085)	0.124 (0.0095)	0.08⋆
VBPR	**1.4** (0.0756)	2.04 (0.0405)	1.0⋆	**1.7** (0.0768)	2.28 (0.0373)	1.0⋆	**0.263** (0.0099)	0.221 (0.0284)	0.992⋆	**0.165** (0.0053)	0.137 (0.0216)	0.976⋆
BiVAECF	**1.44** (0.0628)	1.46 (0.0274)	0.004†	**1.68** (0.061)	1.69 (0.0246)	0.004†	0.226 (0.029)	**0.242** (0.014)	0.644	0.145 (0.0198)	**0.157** (0.0109)	0.587
LayerGCN	**1.68** (0.142)	1.8 (0.197)	0.57⋆	**1.93** (0.135)	2.06 (0.191)	0.587⋆	0.0981 (0.0284)	**0.118** (0.0462)	0.646	0.0703 (0.0246)	**0.0714** (0.0239)	0.281
LGMRec	**0.898** (0.0165)	0.908 (0.0398)	0.0†	**1.09** (0.0104)	1.1 (0.0132)	0.0†	**0.0181** (0.0025)	0.0164 (0.0023)	0.087⋆	**0.0089** (0.0089)	0.0044 (0.0063)	0.0†
DiffMM	**2.59** (0.0097)	3.35 (0.003)	1.0⋆	**2.78** (0.0086)	3.37 (0.0029)	1.0⋆	**0.0248** (0.0013)	0.0047 (0.0011)	1.0⋆	**0.0253** (0.0031)	0.0014 (3.0e-4)	0.0†

Model	Ranking					
	MAP			NDCG@10		
	Cond.	Orig.	p-value	Cond.	Orig.	p-value
MCF	**0.0371** (0.0134)	0.0364 (0.0074)	0.092⋆	**0.0424** (0.0353)	0.04 (0.0306)	0.345⋆
CDL	0.147 (0.0031)	**0.148** (0.0071)	0.0†	0.206 (0.0024)	**0.209** (0.0092)	0.036†
CVAE	**0.157** (0.0066)	0.154 (0.0067)	0.063⋆	**0.219** (0.0071)	0.214 (0.0106)	0.205⋆
VBPR	**0.208** (0.0085)	0.176 (0.0245)	0.972⋆	**0.305** (0.0124)	0.25 (0.0351)	0.998⋆
BiVAECF	0.179 (0.0234)	**0.195** (0.0142)	0.707	0.265 (0.0372)	**0.282** (0.0164)	0.658
LayerGCN	0.0822 (0.0236)	**0.0894** (0.0281)	0.43	0.106 (0.041)	**0.133** (0.0564)	0.671
LGMRec	**0.469** (0.156)	0.417 (0.214)	0.0†	**0.0181** (0.0025)	0.0164 (0.0023)	0.12⋆
DiffMM	**0.568** (0.0088)	0.0049 (0.0011)	1.0⋆	**0.0248** (0.0013)	0.0047 (0.0011)	1.0⋆

Nevertheless, the overall trends and comparisons between the condensed and original high-dimensional item representations remain valid, as the same measurement approach was consistently applied across all experiments.

On all the datasets, using the condensed representations consistently led to significant reductions in resource consumption. On the Movielens dataset (Table 5), the condensed representations induced faster training times, lower average and Peak RAM Memory usage, and lower average and Peak GPU memory usage for all algorithms.

On the larger Amazon Clothing dataset (Table 7), the condensed representations achieved substantial savings in training time, testing time, RAM and GPU memory usage across all algorithms.

Similarly, the results for the Amazon Sports dataset (Table 9) showed the resource efficiency of the condensed representations.

Finally, we obtained consistent outcomes for the largest MuMu dataset (Table 11), with most training times and GPU usage data being lower, and with consistent savings on the RAM memory usage when using the condensed item representations.

It is worth noting that the resource consumption needed to obtain the condensed representations (Table 12) was relatively low compared to the savings achieved in the recommendation process. This is especially true for the largest datasets, where training times and memory and GPU usage for training the recommendation algorithms are more impactful.

Figs. 4–6 allow for a direct assessment of resource usage with the original, high-dimensional embeddings compared to the condensed ones. The values for the condensed embeddings include the additional computational costs associated with the condensation process obtained with MDR, as reported in Table 12. For the train and test time (Fig. 4), the values represent the sum of the original training and testing times plus the extra time required for condensation. The memory usage (Fig. 5) and GPU memory usage (Fig. 6) are calculated as weighted averages, considering both the resource usage for training MDRand the usage during the training and testing phase of the recommendation algorithms. Specifically, if T is the original train + test time, M is the original memory usage (either RAM or GPU), E_t is the time for training MDR, and E_m is the memory usage for training MDR, the reported memory usage is $(M \cdot T + E_m \cdot E_t)/(T + E_t)$. This weighted average reflects the overall resource consumption when using MDR. In contrast, the values for the original, high-dimensional embeddings are reported without

Table 5
Results on the Movielens dataset (Time and Memory Usage). In the round parentheses, we report the standard deviation.

Model	Time				Memory and GPU usage							
	Train (sec)		Test (sec)		Avg Mem (MiB)		Peak Mem (MiB)		Peak GPU Mem (MB)		Avg GPU Mem (MB)	
	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.
MCF	**296**	401	**1**	2	**2,245**	3,658	**2,260**	3,658	-	-	-	-
	(28)	(13)	(0)	(0)	(37)	(36)	(31)	(35)				
CDL	**455**	883	**2**	3	**2,673**	3,578	**2,723**	3,591	**742**	1,017	**736**	1,014
	(23)	(54)	(0)	(0)	(61)	(47)	(9)	(26)	(0)	(0)	(10)	(4)
CVAE	**473**	535	2	**1**	**4,540**	4,866	**4,597**	4,873	**1,176**	1,208	**1,169**	1,207
	(25)	(36)	(0)	(0)	(124)	(27)	(78)	(25)	(0)	(0)	(13)	(0)
VBPR	**23,446**	24,683	**7**	8	**2,522**	3,637	**2,522**	3,637	**813**	1,052	**812**	1,052
	(804)	(1,468)	(1)	(1)	(21)	(18)	(20)	(18)	(0)	(17)	(0)	(17)
BiVAECF	**742**	772	**4**	4	**2,540**	3,639	**2,541**	3,639	**821**	1,107	**820**	1,106
	(19)	(8)	(0)	(0)	(1)	(18)	(1)	(18)	(0)	(0)	(0)	(0)
LayerGCN	**436**	456	6	**6**	**2,355**	2,420	**2,385**	2,442	**317**	318	**312**	313
	(24)	(12)	(0)	(0)	(81)	(75)	(32)	(41)	(0)	(1)	(9)	(9)
LGMRec	**47,341**	47,906	188	**183**	3,677	**3,484**	3,677	**3,489**	881	**836**	879	**835**
	(4,673)	(4,482)	(27)	(43)	(37)	(30)	(37)	(20)	(0)	(0)	(4)	(1)
DiffMM	**15,388**	23,372	**769**	772	**3,069**	4,605	**3,080**	9,081	**1,246**	13,186	**1,245**	13,173
	(575)	(1,230)	(39)	(108)	(44)	(32)	(23)	(187)	(0)	(1,624)	(2)	(1,648)

Table 6
Results on the Clothing dataset. The notation is the same as Table 4.

Model	Error						Relevance					
	MAE			RMSE			Precision@10			Recall@10		
	Cond.	Orig.	p-value	Cond.	Orig.	p-value	Cond.	Orig.	p-value	Cond.	Orig.	p-value
MCF	**0.96**	0.96	0.0†	**0.978**	0.979	0.0†	0.0005	**0.0014**	0.0†	0.0037	**0.0106**	0.0†
	(0.0119)	(0.0024)		(0.0118)	(0.002)		(0)	(0.0004)		(0.0002)	(0.0028)	
CDL	**3.25**	3.26	0.0†	**3.28**	3.28	0.0†	0.002	**0.0021**	0.0†	0.0162	**0.0168**	0.0†
	(0.0075)	(0.0061)		(0.0077)	(0.0062)		(0.0003)	(0.0003)		(0.0023)	(0.0021)	
CVAE	3.18	**3.18**	0.0†	3.21	**3.2**	0.0†	**0.0047**	0.0042	0.0†	**0.037**	0.0335	0.216⋆
	(0.0057)	(0.0221)		(0.0057)	(0.0214)		(0.0014)	(0.0017)		(0.0109)	(0.0136)	
VBPR	1.94	**1.77**	0.793	1.98	**1.84**	0.711	0.0009	**0.0012**	0.0†	0.0071	**0.0096**	0.0†
	(0.177)	(0.0728)		(0.174)	(0.0731)		(0.0001)	(0.0003)		(0.0008)	(0.0021)	
BiVAECF	**3.25**	3.26	0.0†	**3.27**	3.29	0.0†	0.0016	**0.0018**	0.0†	0.0133	**0.0147**	0.0†
	(0.0241)	(0.0051)		(0.0225)	(0.0053)		(0.0003)	(0.0003)		(0.0024)	(0.0021)	
LayerGCN	**3.16**	3.25	0.38⋆	**3.19**	3.27	0.35⋆	0.0008	**0.0009**	0.0†	0.0066	**0.0068**	0.0†
	(0.113)	(0.0357)		(0.11)	(0.0346)		(0.0003)	(0.0003)		(0.0024)	(0.0026)	
LGMRec	1.33	**0.787**	0.895	1.35	**0.814**	0.895	**0.0007**	0.0004	0.0†	**0.001**	0.0009	0.0†
	(0.792)	(0.1)		(0.794)	(0.0954)		(6.0e-4)	(3.0e-4)		(3.0e-4)	(3.0e-4)	
DiffMM	3.3	3.3	0.0†	3.32	3.32	0.0†	0.0045	**0.0048**	0.0†	0.0039	**0.0042**	0.0†
	(0.0063)	(0.0063)		(0.0064)	(0.0064)		(0.001)	(0.0012)		(0.0012)	(0.0015)	

Model	Ranking					
	MAP			NDCG@10		
	Cond.	Orig.	p-value	Cond.	Orig.	p-value
MCF	0.0027	**0.0059**	0.0†	0.002	**0.0057**	0.0†
	(3.0e-4)	(0.0012)		(2.0e-4)	(0.0016)	
CDL	0.0092	**0.0095**	0.0†	0.0088	**0.0093**	0.0†
	(9.0e-4)	(0.0014)		(0.0013)	(0.0017)	
CVAE	**0.0193**	0.0166	0.023†	**0.0213**	0.0184	0.062⋆
	(0.0055)	(0.0063)		(0.0067)	(0.0076)	
VBPR	0.0043	**0.0054**	0.0†	0.0036	**0.0048**	0.0†
	(2.0e-4)	(7.0e-4)		(3.0e-4)	(0.0011)	
BiVAECF	0.0067	**0.0072**	0.0†	0.0066	**0.0073**	0.0†
	(0.0012)	(8.0e-4)		(0.0013)	(0.0011)	
LayerGCN	0.0034	0.0034	0.0	0.0026	**0.0033**	0.0†
	(8.0e-4)	(0.0011)		(9.0e-4)	(0.0012)	
LGMRec	**0.187**	0.0732	0.0†	**0.0007**	0.0004	0.0†
	(0.185)	(0.146)		(0.0006)	(0.0003)	
DiffMM	0.54	**0.541**	0.0†	0.0045	**0.0048**	0.0†
	(0.0097)	(0.0095)		(0.001)	(0.0012)	

Table 7
Results on the clothing dataset (Time and Memory Usage). In the round parentheses, we report the standard deviation.

Model	Time				Memory and GPU usage							
	Train (sec)		Test (sec)		Avg Mem (MiB)		Peak Mem (MiB)		Peak GPU Mem (MB)		Avg GPU Mem (MB)	
	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.
MCF	**397**	413	**16**	20	**889**	1,581	**911**	1,603	-	-	-	-
	(37)	(17)	(0)	(0)	(13)	(3)	(14)	(3)				
CDL	**12,992**	17,874	**35**	37	**4,130**	5,380	**4,207**	5,620	**1,165**	1,211	**1,144**	1,210
	(677)	(986)	(1)	(1)	(39)	(27)	(28)	(38)	(62)	(0)	(59)	(0)
CVAE	**11,768**	13,869	**37**	39	**4,246**	5,277	**4,360**	5,402	1,217	**1,211**	1,217	**1,211**
	(562)	(768)	(3)	(1)	(9)	(9)	(9)	(45)	(0)	(0)	(0)	(0)
VBPR	**18,239**	20,479	**183**	183	**4,095**	5,121	**4,095**	5,123	**1,325**	1,402	**1,324**	1,400
	(861)	(926)	(5)	(9)	(3)	(4)	(3)	(4)	(0)	(10)	(0)	(10)
BiVAECF	**11,550**	12,092	**21**	21	**4,171**	5,102	**4,192**	5,105	**1,476**	1,776	**1,475**	1,776
	(357)	(768)	(1)	(1)	(4)	(15)	(30)	(14)	(1)	(1)	(2)	(1)
LayerGCN	**2,525**	2,647	**38**	39	**3,071**	4,223	**3,143**	4,223	**1,474**	1,497	**1,426**	1,496
	(235)	(89)	(1)	(1)	(191)	(3)	(54)	(3)	(9)	(0)	(91)	(1)
LGMRec	**36,516**	39,003	**216**	257	**3,553**	4,490	**3,558**	4,548	**1,189**	3,381	**1,183**	3,381
	(3,365)	(2,103)	(50)	(45)	(64)	(0)	(55)	(33)	(0)	(0)	(11)	(0)
DiffMM	**27,583**	29,349	100	**89**	4,494	**3,644**	5,218	**4,374**	3,383	**3,163**	3,383	**3,161**
	(1,957)	(989)	(26)	(23)	(0)	(31)	(20)	(29)	(2)	(266)	(2)	(268)

Table 8
Results on the Sports dataset. The notation is the same as Table 4.

Model	Error						Relevance					
	MAE			RMSE			Precision@10			Recall@10		
	Cond.	Orig.	p-value	Cond.	Orig.	p-value	Cond.	Orig.	p-value	Cond.	Orig.	p-value
MCF	**0.721**	0.763	0.0†	**0.778**	0.821	0.0†	0.0002	**0.0004**	0.0†	0.0009	**0.0019**	0.0†
	(0.0296)	(0.0123)		(0.0279)	(0.01)		(0.0001)	(0.0003)		(0.0007)	(0.0012)	
CDL	**3.15**	3.16	0.0†	**3.22**	3.23	0.0†	**0.0102**	0.0099	0.0†	**0.0503**	0.0487	0.0†
	(0.0055)	(0.0055)		(0.0051)	(0.0049)		(0.0003)	(0.0002)		(0.0015)	(0.0012)	
CVAE	**3.08**	3.08	0.0†	**3.15**	3.16	0.0†	**0.0106**	0.0105	0.0†	**0.0521**	0.0516	0.0†
	(0.0121)	(0.0089)		(0.0112)	(0.0088)		(0.0007)	(0.0005)		(0.0031)	(0.0029)	
VBPR	1.64	**1.22**	1.0	1.81	**1.41**	1.0	0.0034	**0.0041**	0.0†	0.0165	**0.0189**	0.0†
	(0.159)	(0.0687)		(0.152)	(0.0771)		(0.0004)	(0.0004)		(0.0023)	(0.0021)	
BiVAECF	3.37	**3.37**	0.0†	3.43	**3.43**	0.0†	0.0082	**0.0085**	0.0†	0.0396	**0.0406**	0.0†
	(0.0046)	(0.004)		(0.0041)	(0.0036)		(0.0004)	(0.0001)		(0.002)	(0.001)	
LayerGCN	**3.34**	3.35	0.0†	**3.4**	3.41	0.0†	0.0003	**0.0004**	0.0†	0.0013	**0.0019**	0.0†
	(0.0508)	(0.0238)		(0.0464)	(0.0222)		(0.0001)	(0.0002)		(0.0006)	(0.0008)	
LGMRec	**0.685**	0.854	0.654⋆	**0.759**	0.921	0.69⋆	0.0001	**0.0004**	0.0†	0.0002	**0.0004**	0.0†
	(0.0524)	(0.284)		(0.0347)	(0.266)		(0.0e+00)	(3.0e-4)		(1.0e-4)	(2.0e-4)	
DiffMM	3.38	3.38	0.0†	3.44	**3.44**	0.0†	0.0054	**0.0056**	0.0†	0.0065	**0.0067**	0.0†
	(0.0045)	(0.0041)		(0.004)	(0.0037)		(6.0e-4)	(5.0e-4)		(9.0e-4)	(5.0e-4)	

Model	Ranking					
	MAP			NDCG@10		
	Cond.	Orig.	p-value	Cond.	Orig.	p-value
MCF	8.0e-4	**0.0013**	0.0†	5.0e-4	**0.0011**	0.0†
	(3.0e-4)	(6.0e-4)		(4.0e-4)	(7.0e-4)	
CDL	**0.0264**	0.0252	0.0†	**0.0318**	0.0303	0.0†
	(8.0e-4)	(7.0e-4)		(0.001)	(9.0e-4)	
CVAE	**0.0276**	0.0275	0.0†	**0.0334**	0.0333	0.0†
	(0.0013)	(0.0015)		(0.0019)	(0.0018)	
VBPR	0.0091	**0.0101**	0.0†	0.0099	**0.0113**	0.0†
	(0.0011)	(0.0013)		(0.0014)	(0.0017)	
BiVAECF	0.0211	**0.022**	0.0†	0.0249	**0.0259**	0.0†
	(0.001)	(5.0e-4)		(0.0014)	(6.0e-4)	
LayerGCN	9.0e-4	**0.0011**	0.0†	8.0e-4	**0.0011**	0.0†
	(3.0e-4)	(2.0e-4)		(3.0e-4)	(3.0e-4)	
LGMRec	0.0335	**0.185**	0.0†	0.0001	**0.0004**	0.0†
	(0.0671)	(0.232)		(0)	(0.0003)	
DiffMM	0.551	**0.553**	0.0†	0.0054	**0.0056**	0.0†
	(0.0147)	(0.018)		(0.0006)	(0.0005)	

Table 9
Results on the sports dataset (Time and Memory Usage). In the round parentheses, we report the standard deviation.

Model	Time				Memory and GPU usage							
	Train (sec)		Test (sec)		Avg Mem (MiB)		Peak Mem (MiB)		Peak GPU Mem (MB)		Avg GPU Mem (MB)	
	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.
MCF	**1,663** (87)	1,717 (80)	**249** (6)	249 (5)	**4,470** (61)	5,537 (43)	**4,472** (58)	5,558 (45)	-	-	-	-
CDL	**58,693** (4,578)	74,389 (5,305)	249 (8)	**248** (1)	**5,606** (91)	9,452 (114)	**5,699** (74)	10,149 (141)	**1,211** (0)	1,225 (0)	**1,210** (0)	1,224 (0)
CVAE	**53,520** (1,352)	54,382 (3,566)	**249** (4)	249 (1)	**5,719** (106)	8,887 (42)	**5,891** (86)	9,524 (60)	**1,465** (0)	1,481 (0)	**1,455** (19)	1,477 (6)
VBPR	**77,320** (4,576)	82,316 (3,301)	**1,562** (25)	1,591 (67)	**5,840** (22)	8,694 (18)	**5,840** (21)	8,707 (19)	**1,595** (0)	1,932 (0)	**1,594** (1)	1,931 (3)
BiVAECF	**37,617** (2,444)	42,342 (1,803)	**141** (3)	143 (2)	**4,877** (245)	8,021 (35)	**4,969** (134)	8,050 (29)	**400** (200)	497 (6)	**400** (200)	497 (6)
LayerGCN	**2,025** (200)	4,144 (164)	**144** (2)	289 (1)	**3,604** (176)	7,471 (29)	**3,619** (154)	7,471 (30)	**2,491** (116)	2,599 (0)	**2,415** (173)	2,599 (0)
LGMRec	**113,324** (4,011)	159,189 (6,946)	**2,190** (445)	2,526 (319)	**5,002** (145)	8,200 (12)	**5,004** (141)	8,518 (10)	**1,227** (0)	2,517 (0)	**1,224** (4)	2,514 (4)
DiffMM	20,164 (4,044)	**20,127** (3,688)	**683** (155)	842 (174)	**4,995** (151)	8,208 (4)	**7,968** (246)	11,334 (130)	9,229 (0)	**0** (0)	**9,093** (271)	9,229 (0)

Table 10
Results on the MuMu dataset. The notation is the same as Table 4.

Model	Error						Relevance					
	MAE			RMSE			Precision@10			Recall@10		
	Cond.	Orig.	p-value	Cond.	Orig.	p-value	Cond.	Orig.	p-value	Cond.	Orig.	p-value
MCF	0.161 (0.0035)	**0.16** (0.007)	0.0†	**0.171** (0.0037)	0.172 (0.0031)	0.0†	**0.116** (0.0191)	0.115 (0.0195)	0.233⋆	0.589 (0.0818)	**0.593** (0.0846)	0.447
CDL	**2.56** (0.0131)	2.6 (0.0178)	0.0†	**2.6** (0.0128)	2.63 (0.0177)	0.0†	**0.0756** (0.0023)	0.0741 (0.0018)	0.0†	**0.511** (0.0174)	0.5 (0.0116)	0.465⋆
CVAE	**2.18** (0.0739)	2.21 (0.0861)	0.099⋆	**2.22** (0.074)	2.25 (0.0852)	0.083⋆	0.0787 (0.014)	**0.0862** (0.0044)	0.331	0.53 (0.0965)	**0.582** (0.0288)	0.794
VBPR	1.58 (0.408)	**0.921** (0.0476)	1.0	1.62 (0.403)	**0.99** (0.0511)	1.0	**0.0614** (0.0109)	0.0517 (0.009)	0.491⋆	**0.427** (0.075)	0.357 (0.0635)	0.931⋆
BiVAECF	3.3 (0.0038)	**3.29** (0.0032)	0.0†	3.32 (0.0038)	**3.31** (0.0032)	0.0†	**0.0508** (0.0027)	0.0484 (0.0035)	0.0†	**0.346** (0.0186)	0.329 (0.0235)	0.653⋆
LayerGCN	**3.25** (0.0654)	3.26 (0.0601)	0.009†	**3.28** (0.0627)	3.29 (0.0561)	0.002†	0.0013 (0.0004)	**0.0019** (0.0007)	0.0†	0.009 (0.0028)	**0.0132** (0.0055)	0.017†
LGMRec	2.17 (0.121)	**2.07** (0.101)	0.466	2.2 (0.118)	**2.1** (0.0976)	0.465	**0.0729** (0.0649)	0.0668 (0.0589)	0.197⋆	**0.0186** (0.0162)	0.0172 (0.0147)	0.512⋆
DiffMM	3.35 (0.003)	**3.35** (0.0042)	0.0†	3.37 (0.0029)	**3.37** (0.004)	0.0†	**0.0047** (0.0011)	0.0039 (9.0e-4)	0.0†	**0.0014** (3.0e-4)	0.0012 (2.0e-4)	0.0†

Model	Ranking					
	MAP			NDCG@10		
	Cond.	Orig.	p-value	Cond.	Orig.	p-value
MCF	0.453 (0.103)	**0.453** (0.102)	0.455	**0.49** (0.102)	0.485 (0.101)	0.495⋆
CDL	**0.207** (0.0067)	0.203 (0.0045)	0.049†	**0.282** (0.0099)	0.277 (0.0066)	0.204⋆
CVAE	0.214 (0.0366)	**0.233** (0.0112)	0.687	0.292 (0.0536)	**0.32** (0.0159)	0.747
VBPR	**0.177** (0.0306)	0.153 (0.0242)	0.782⋆	**0.237** (0.0421)	0.201 (0.0344)	0.848⋆
BiVAECF	**0.135** (0.0083)	0.131 (0.009)	0.143⋆	**0.185** (0.0101)	0.178 (0.0129)	0.342⋆
LayerGCN	0.0045 (9.0e-4)	**0.0064** (0.0025)	0.0†	0.0048 (0.0014)	**0.0073** (0.0034)	0.0†
LGMRec	**0.058** (0.0481)	0.055 (0.0447)	0.389⋆	**0.0729** (0.0649)	0.0668 (0.0589)	0.449⋆
DiffMM	**0.0049** (0.0011)	0.004 (0.0012)	0.0†	**0.0047** (0.0011)	0.0039 (0.0009)	0.0†

Table 11
Results on the MuMu dataset (Time and Memory Usage). In the round parentheses, we report the standard deviation.

Model	Time				Memory and GPU usage							
	Train (sec)		Test (sec)		Avg Mem (MiB)		Peak Mem (MiB)		Peak GPU Mem (MB)		Avg GPU Mem (MB)	
	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.	Cond.	Orig.
MCF	**7,104**	7,182	**6,396**	6,449	**4,083**	4,110	**4,519**	4,527	-	-	-	-
	(576)	(670)	(175)	(58)	(330)	(375)	(338)	(380)				
CDL	**102,402**	130,470	201	**199**	**3,252**	10,856	**3,427**	11,332	833	833	**823**	824
	(5,568)	(4,323)	(2)	(2)	(162)	(107)	(152)	(85)	(0)	(0)	(19)	(17)
CVAE	**90,497**	90,782	199	**199**	**3,517**	10,649	**3,730**	11,067	833	833	833	833
	(4,501)	(6,382)	(2)	(1)	(106)	(32)	(104)	(103)	(0)	(0)	(0)	(0)
VBPR	**53,022**	58,994	**205**	205	**2,535**	10,144	**2,541**	10,148	**562**	815	**558**	814
	(3,649)	(4,909)	(1)	(2)	(61)	(34)	(60)	(31)	(11)	(0)	(12)	(0)
BiVAECF	**62,575**	70,450	150	**150**	**3,211**	9,477	**3,245**	9,558	**532**	773	**532**	772
	(4,773)	(3,371)	(1)	(1)	(176)	(47)	(122)	(36)	(0)	(0)	(0)	(1)
LayerGCN	**3,691**	3,969	**146**	151	**3,201**	9,648	**3,207**	9,653	**3,249**	3,856	**3,127**	3,746
	(407)	(407)	(5)	(5)	(226)	(67)	(215)	(60)	(264)	(145)	(355)	(277)
LGMRec	**22,011**	37,189	**2,436**	3,066	**4,422**	7,304	**4,428**	7,304	**1,522**	13,999	**1,500**	13,999
	(2,468)	(9,247)	(436)	(315)	(160)	(0)	(151)	(0)	(38)	(0)	(57)	(0)
DiffMM	**23,372**	24,194	**772**	935	**4,605**	7,325	**9,081**	11,556	**13,186**	13,999	**13,173**	13,999
	(1,230)	(3,626)	(108)	(183)	(32)	(2)	(187)	(141)	(1,624)	(0)	(1,648)	(0)

Table 12
Resource consumption statistics for item condensation on different datasets. The results refer to training a single model on 5 folds.

Dataset	Movielens	Clothing	Sports	MuMu
Avg. memory usage (MiB)	1,390	4,044	10,012	7,352
Peak RAM Memory usage (MiB)	1,416	4,123	10,098	8,238
Avg. GPU usage (MB)	3,197	14,811	9,200	3,101
Peak GPU usage (MB)	3,217	14,854	9,209	3,111
Duration (s)	8	51	362	158

any adjustments, as they do not require any pre-processing. We notice that for what concerns Train + Test Time (Fig. 4) the results are slightly lower when using the condensed representation in the recommendation algorithms across the different datasets, with the benefits of using the condensed representation being more pronounced for the Amazon clothing and Amazon sports datasets. Conversely, looking at Average Memory Usage (MiB) (Fig. 5) and the Average GPU Memory Usage (MB) (Fig. 6) we notice a significant reduction in resource consumption.

9. Discussion

The results presented in Section 8 reveal interesting patterns that bridge our two research questions. These findings paint a promising picture of the impact of our fusion method, demonstrating its ability to enhance recommendation quality and reduce computational costs simultaneously. As such, they enable us to answer the research questions.

RQ1. Do the condensed item representations obtained through the proposed fusion method (MDR) preserve recommendation performance compared to the original high-dimensional versions?. We observed that the condensed item representations obtained through our proposed fusion method not only preserve but often enhance the recommendation performance across diverse datasets and algorithms. On the four datasets we considered (Movielens, Amazon clothing, Amazon sports, and MuMu), diverse recommender systems achieved comparable or even better recommendation performance when using the condensed item representations than the original high-dimensional ones. For recommendation performance evaluation, we used recommendation metrics measuring errors, relevance, and ranking.

There were a few exceptions where the original representations prevailed, most notably with the VBPR algorithm on the MuMu, Amazon clothing and sports datasets regarding error metrics. However, the overall trend strongly indicates that the condensed representations produced by MDR can effectively capture the salient information from the original high-dimensional multimodal item features without compromising recommendation quality. Especially on the Movielens and MuMu datasets, the condensed representations frequently led to improved recommendation performance. This finding directly addresses our first research question and underscores the effectiveness of MDR in capturing the salient information from the original high-dimensional multimodal features.

RQ2. Can the condensed representations obtained with MDRhelp save computational resources in terms of RAM and GPU memory usage, and training and test times?. The described recommendation performance improvement is accompanied by significant reductions in resource consumption, as evidenced by the faster training times, lower RAM usage, and reduced GPU utilization induced on algorithms by the condensed representations. This also holds in most cases when we include the resource consumption related to the execution of the condensation model.

This observation ties into our second research question and highlights the potential of our method to improve the efficiency and sustainability of multimodal recommender systems.

These efficiency gains have important implications for the sustainability, accessibility, and development of multimodal recommender systems. MDR can help lower the energy consumption and carbon footprint associated with training and deploying these models. Moreover, the improved efficiency makes it more feasible for organizations with limited computational resources to leverage multimodal item features in their recommender systems.

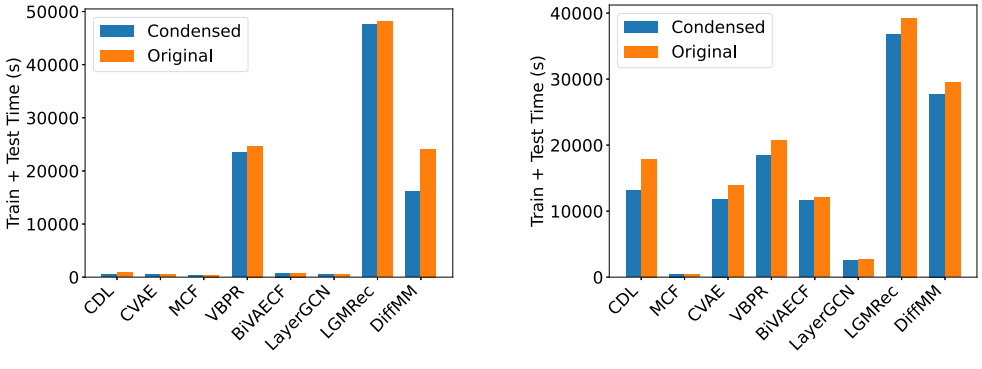
In a real-world production setting, the faster training times enabled by the condensed representations would also allow for more frequent model updates as new user interaction data becomes available. This could potentially lead to more responsive and up-to-date recommendations.

The consistent resource savings across datasets and algorithms suggest that the benefits of MDR are not limited to specific settings of algorithms and datasets but can be generalized to various recommendation scenarios.

10. Limitations and future work

Despite the effort we put into this work, there are some limitations to be resolved with further research.

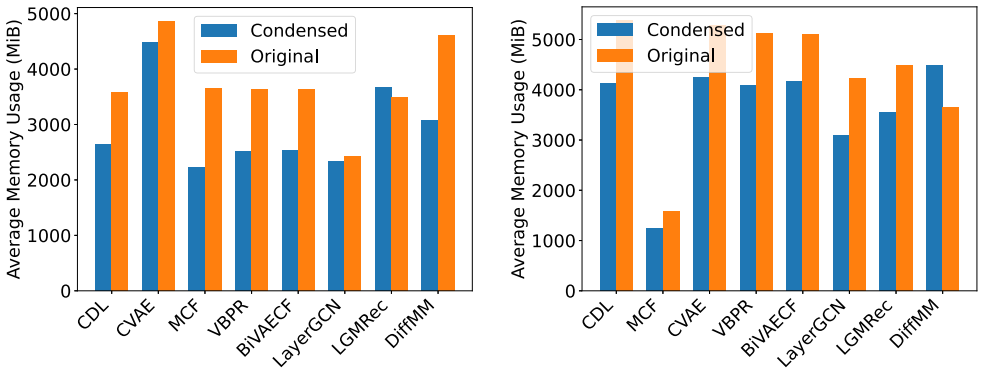
One limitation is that, when summing the training times of the fastest recommendation algorithms with those of our condensation model, we obtained superior results with the original representation. However, on



(a) MovieLens dataset (b) Clothing dataset

(c) Sports dataset (d) Mumu dataset

Fig. 4. Comparison of Train + Test Time (s) across different datasets.



(a) MovieLens dataset (b) Clothing dataset

(c) Sports dataset (d) Mumu dataset

Fig. 5. Comparison of average memory usage (MiB) across different datasets.

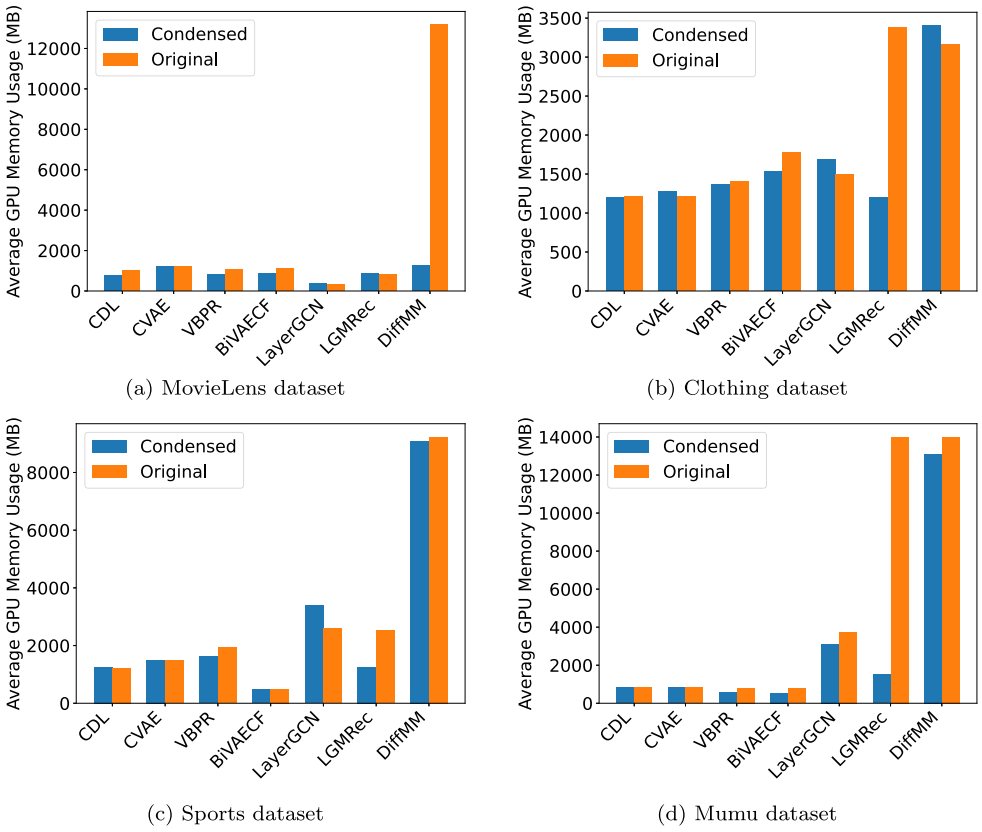


Fig. 6. Comparison of average GPU Memory Usage (MB) across different datasets.

the Movielens dataset, we found that the condensation was successful even with a dramatically smaller instances. Thus, efficiently sampling the dataset for the training of the condensation model seems a promising approach to mitigate this issue. We plan to develop and test an efficient sampling strategy on a set of new datasets to validate our hypothesis.Furthermore, in our work, we evaluated resource usage by monitoring GPU memory consumption as well as training and testing times. For future work, we aim to gather more comprehensive data by including metrics such as the number of trainable parameters and the speed of convergence. This will allow us to present more detailed and insightful results.

In our future work, we also plan to test the proposed model's ability to integrate diverse, complementary information (e.g., in a food recommender system, the information of how food is produced) from multiple modalities rather than classic multimodal information (e.g., nutritional profile, images of the food, etc.), aiming to obtain a unified and coherent representation of items.

Finally, comparing the recommendations produced using high-dimensional item representations and condensed ones through a user study would be beneficial to investigate further the impact of using MDR.

11. Conclusion

In this work, we proposed MDR, a fusion method based on attention bottlenecks to obtain condensed multimodal item representations for recommender systems. MDR leverages a Transformer autoencoder architecture to learn compact representations that preserve the salient information from the original high-dimensional features. Experiments on four datasets demonstrated that diverse recommender systems achieve comparable or even higher recommendation quality when using item representations produced by our fusion method compared to using the original item representations. Moreover, the condensed representations support significant reductions in resource usage in most cases. These findings highlight the potential of MDR to enhance the efficiency and environmental sustainability of multimodal recommenders without compromising their capability to generate good recommendations.

CRediT authorship contribution statement

Angelo Geninatti Cossatin: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing; **Noemi Mauro:** Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Supervision, Validation, Writing – original draft, Writing – review & editing.

Data availability

We add the link in the paper.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Boxplots correlating each hyperparameter with the reconstruction error

The following boxplots visualize the impact of varying each hyperparameter of the MDR model on the resulting reconstruction error. The reconstruction error measures how well the model is able to reconstruct the original high-dimensional item representations from the condensed representations it learns. A lower reconstruction error indicates the model is better preserving the salient information while reducing dimensionality.

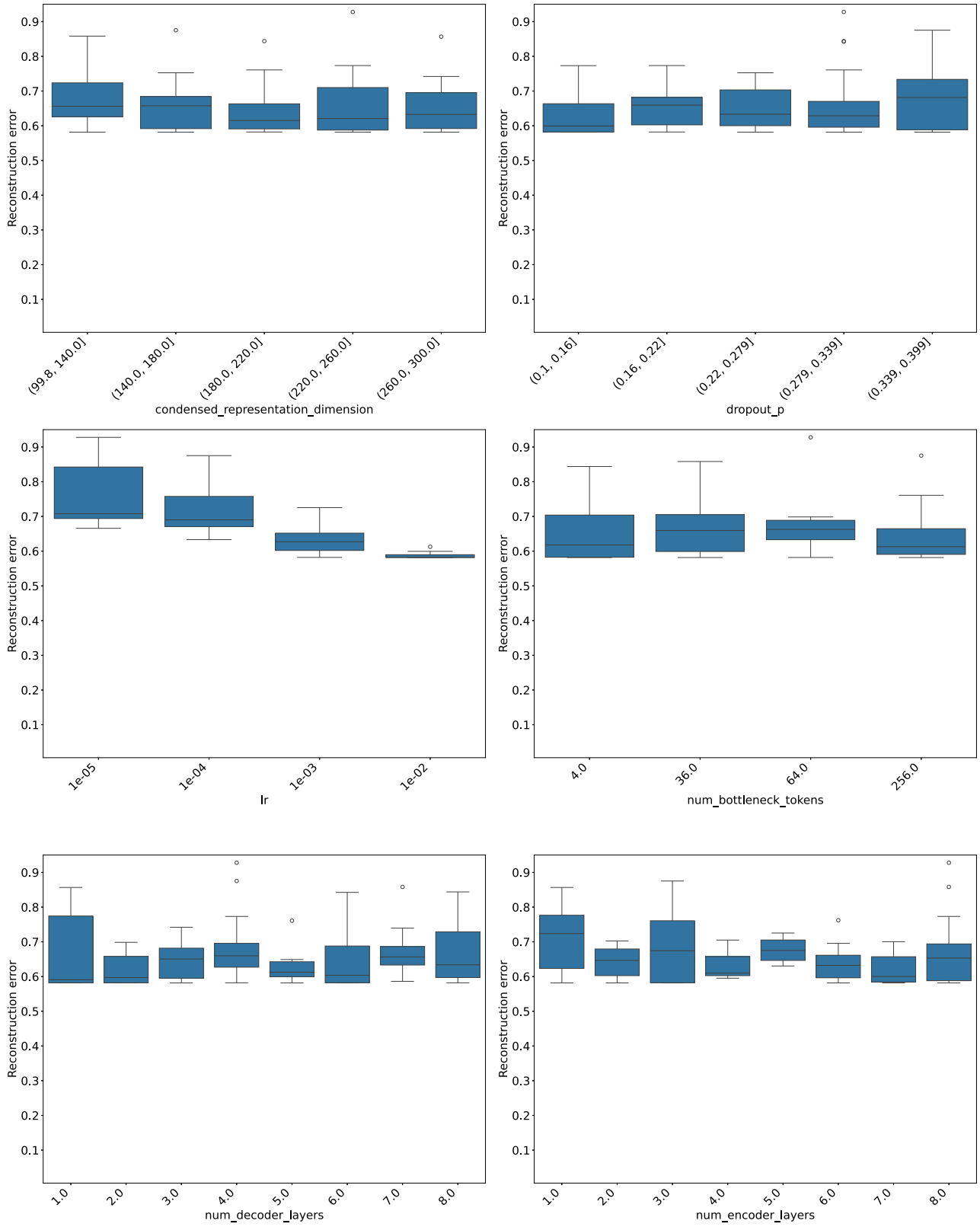


Fig. A.1. Boxplot for the Movielens dataset. The learning rate (lr) has the strongest impact, with higher learning rates leading to lower reconstruction error. The model is relatively robust to variations in the other hyperparameters like the number of encoder layers, decoder layers, bottleneck tokens, dropout probability, and size of the condensed representation.

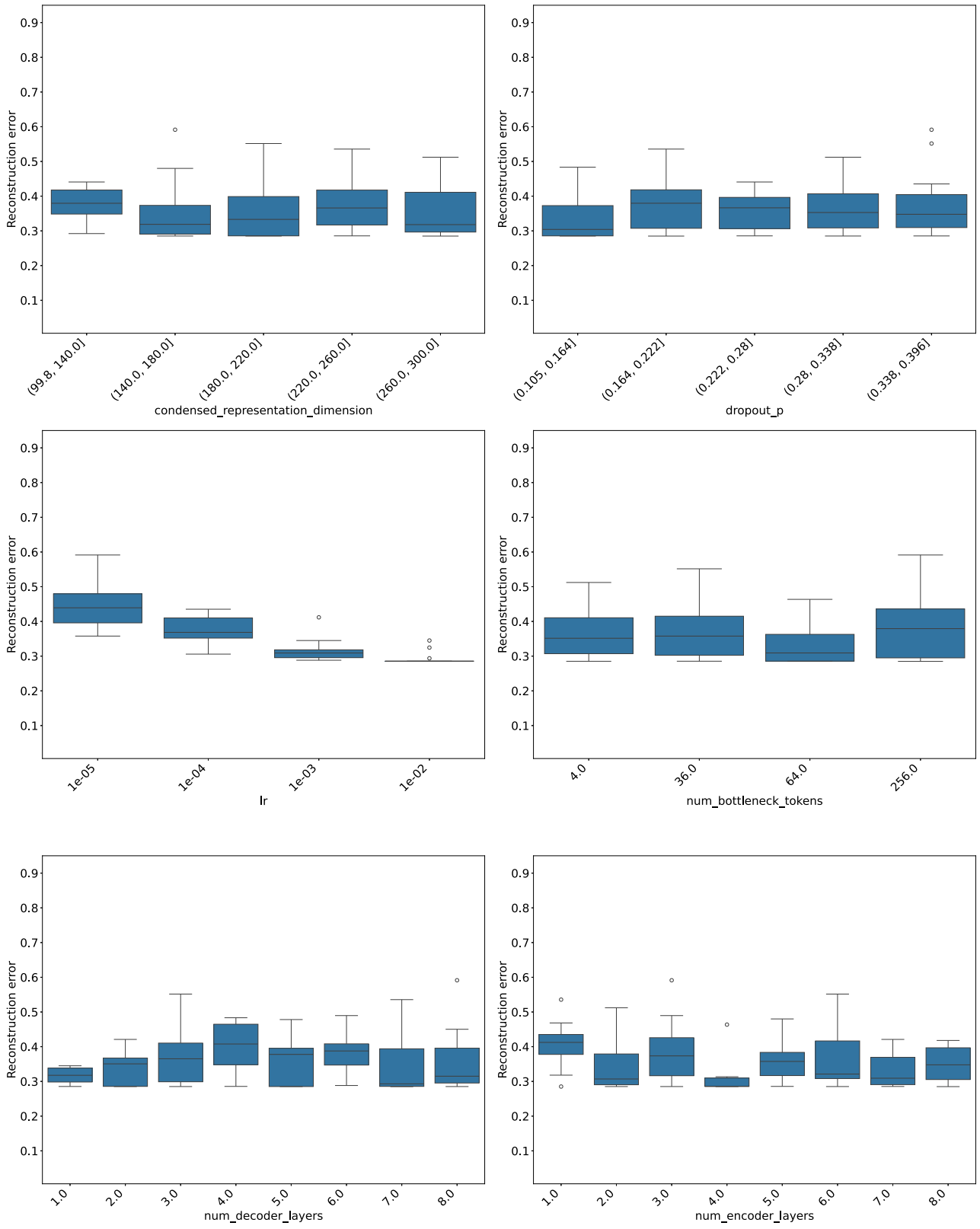


Fig. A.2. Boxplot for the amazon clothing dataset. The learning rate (lr) exhibits the most significant influence, with higher values resulting in lower reconstruction errors. The remaining hyperparameters have a more limited impact on the model's performance.

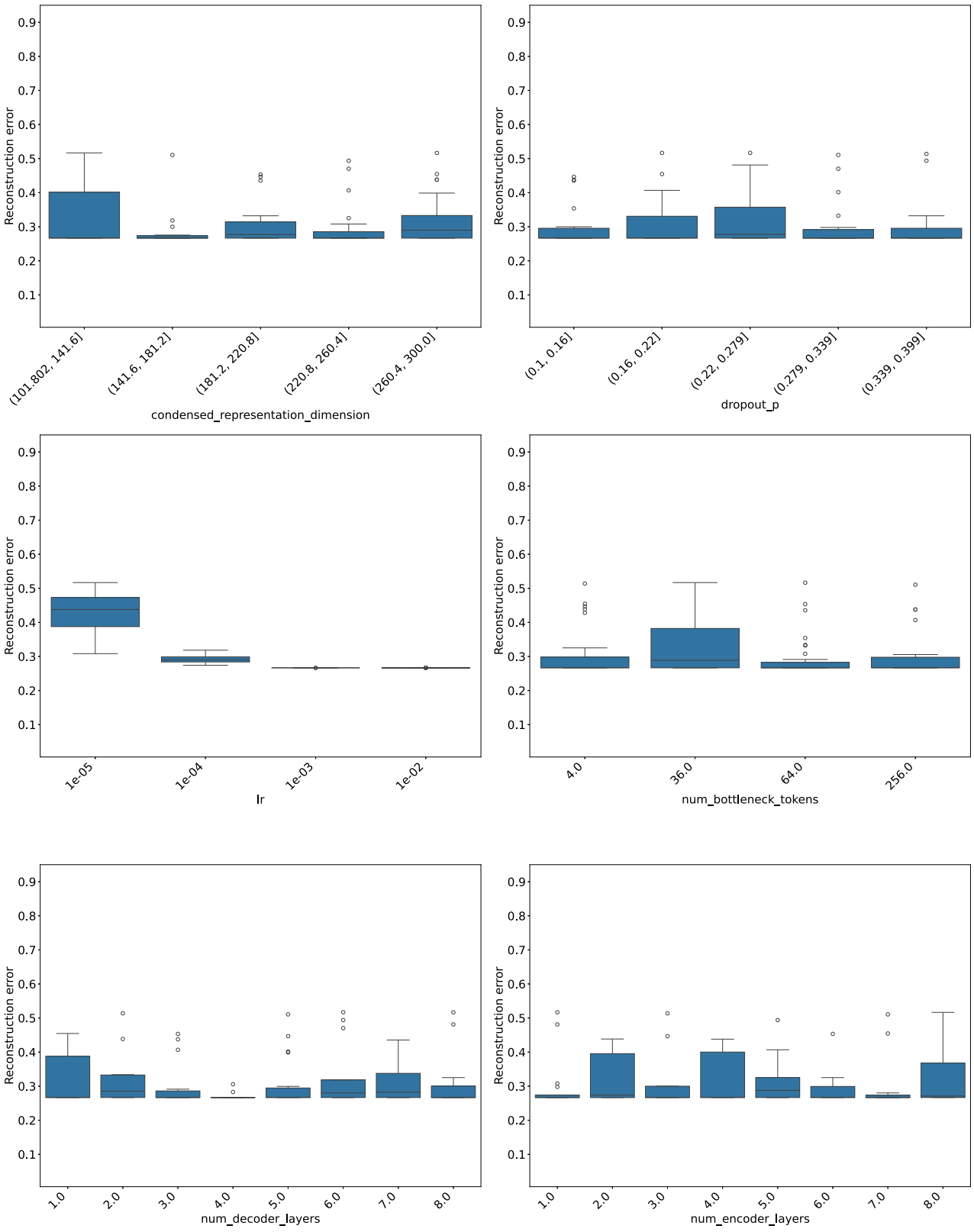


Fig. A.3. Boxplot for the amazon sports dataset. Once again, the learning rate (lr) emerges as the most influential factor, with higher rates leading to lower reconstruction errors. The model's performance is less sensitive to changes in the other hyperparameters.

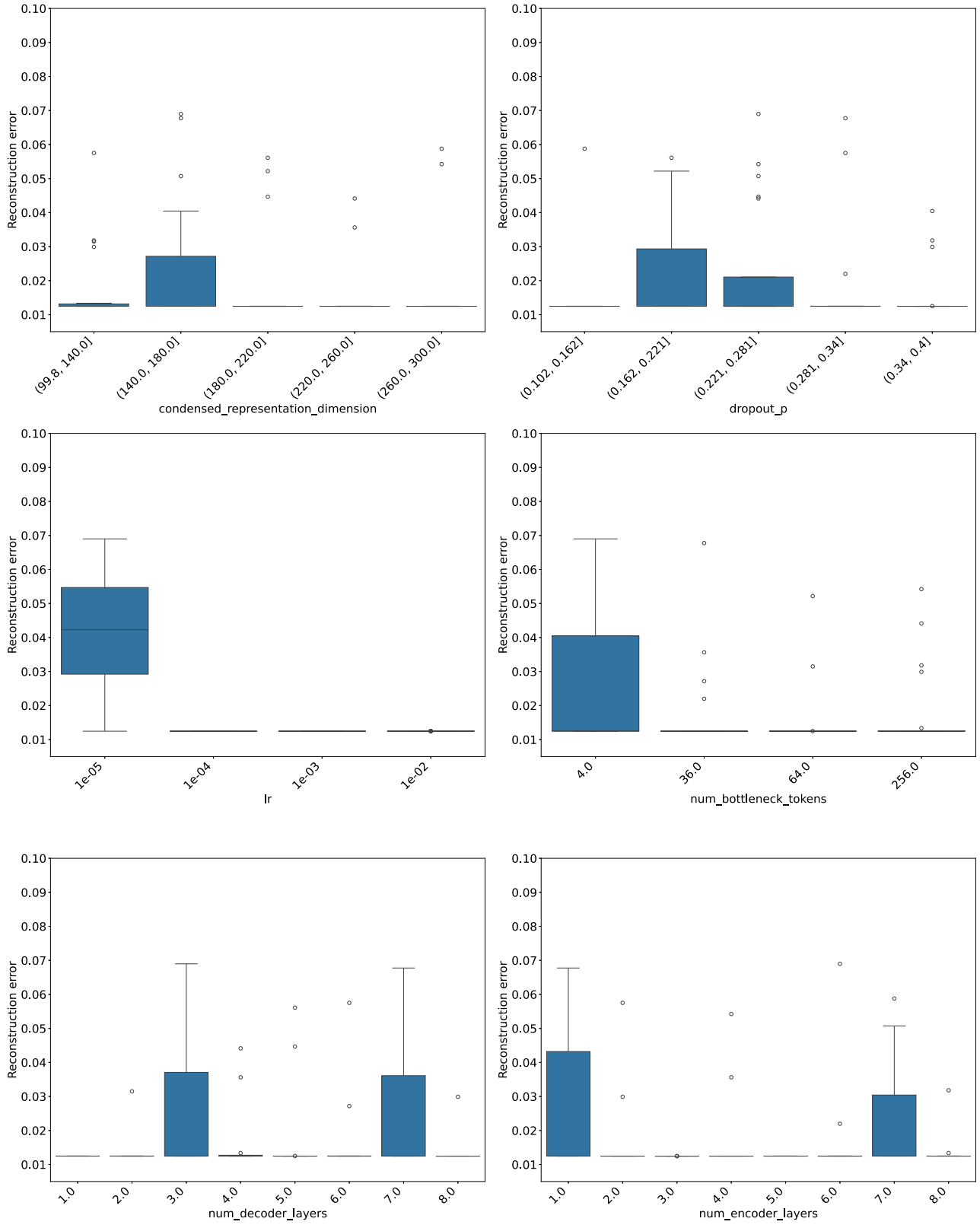
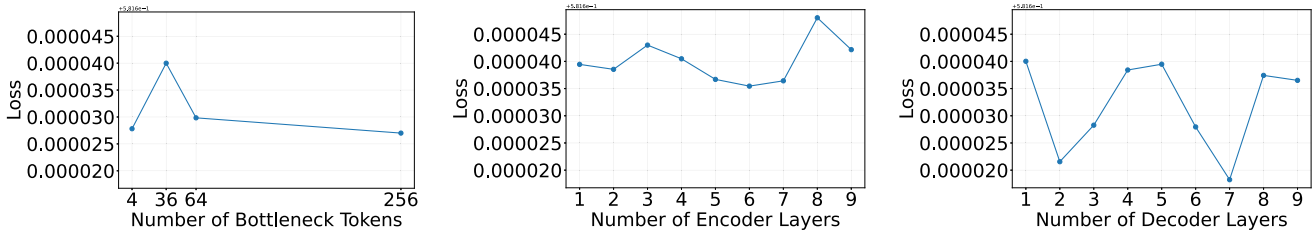


Fig. A.4. Boxplot for the MuMu dataset. Note that the y-axis scale is narrower compared to the other datasets, reflecting MDR's increased robustness to hyperparameter variations as the dataset size grows. Nonetheless, the learning rate (lr) still has the most noticeable impact on performance, with higher values yielding better results (lower reconstruction errors). The influence of the remaining hyperparameters is less pronounced.

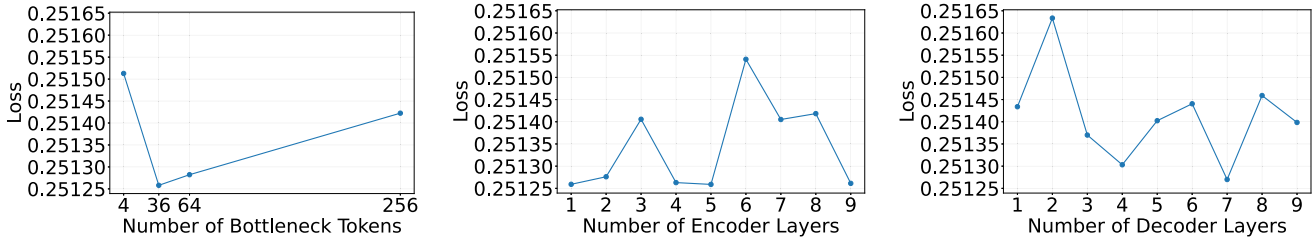
Appendix B. Ablation study for salient hyperparameters

The following line plots visualize the impact of varying the number of bottleneck tokens, encoder and decoder layers on the best configuration. The loss corresponds to the reconstruction error.



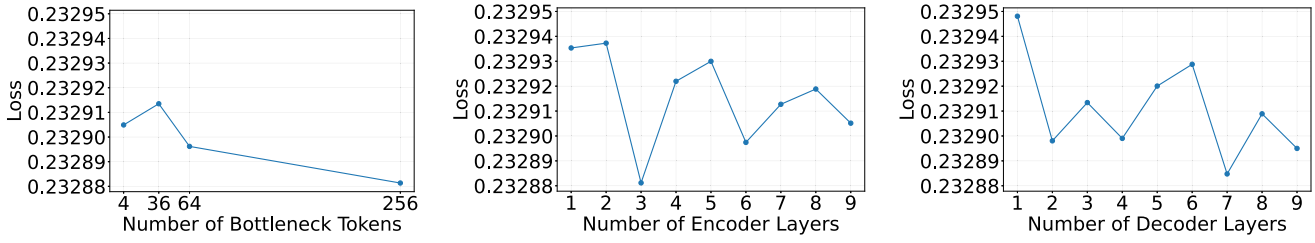
(a) Bottleneck Tokens (b) Encoder Layers (c) Decoder Layers

Fig. B.1. Loss curves for different architectural configurations on MovieLens dataset.



(a) Bottleneck Tokens (b) Encoder Layers (c) Decoder Layers

Fig. B.2. Loss curves for different architectural configurations on Amazon Clothing dataset.



(a) Bottleneck Tokens (b) Encoder Layers (c) Decoder Layers

Fig. B.3. Loss curves for different architectural configurations on Amazon Sports dataset.

References

GroupLens Research (2015). MovieLens 100K dataset. GroupLens Research. https://grouplens.org/datasets/movielens/100k/.

Ba, Q., Li, X., & Bai, Z. (2013). Clustering collaborative filtering recommendation system based on SVD algorithm. In *IEEE 4th International conference on software engineering and service science* (pp. 963–967). https://doi.org/10.1109/ICSESS.2013.6615466

Belsley, D. A., Kontoghiorghes, E. J., & KontoghiorghesE. J (2009). Handbook of computational econometrics. Oxford: Wiley.

Bracher, C., Heinz, S., & Vollgraf, R. (2016). Fashion DNA: Merging content and sales data for recommendation and article mapping. In *"Machine learning meets fashion", kdd 2016*. http://arxiv.org/abs/1609.02489.

Chen, X., Chen, H., Xu, H., Zhang, Y., Cao, Y., Qin, Z., & Zha, H. (2019). Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval Paris France* (pp. 765–774). ACM. https://doi.org/10.1145/3331184.3331254

Deldjoo, Y., Constantin, M. G., Eghbal-Zadeh, H., Ionescu, B., Schedl, M., & Cremonesi, P. (2018). Audio-visual encoding of multimedia content for enhancing movie recommendations. In *Proceedings of the 12th ACM conference on recommender systems, ACM, Vancouver British Columbia Canada* (pp. 455–459). https://doi.org/10.1145/3240323.3240407

Guan, X., Li, C. T., & Guan, Y. (2017). Matrix factorization with rating completion: An enhanced SVD model for collaborative filtering recommender systems. *IEEE Access*, *5*, 27668–27678. https://doi.org/10.1109/ACCESS.2017.2772226

Guo, Z., Li, J., Li, G., Wang, C., Shi, S., & Ruan, B. (2024). LGMREC: Local and global graph learning for multimodal recommendation. *Proceedings of the AAAI conference on artificial intelligence*, *38*, 8454–8462. https://doi.org/10.1609/aaai.v38i8.28688

Han, T., Wang, P., Niu, S., & Li, C. (2022). Modality matches modality: Pretraining modality-disentangled item representations for recommendation. In *Proceedings of the ACM web conference 2022, ACM, virtual Event Lyon France* (pp. 2058–2066). https://doi.org/10.1145/3485447.3512079

He, R., & Mcauley, J. (2016a). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th international conference on world wide web, international world wide web conferences steering committee, republic and canton of Geneva* (pp. 507–517). https://doi.org/10.1145/2872427.2883037

He, R., & Mcauley, J. (2016b). VBPR: Visual Bayesian personalized ranking from implicit feedback. In *Proceedings of the thirtieth AAAI conference on artificial intelligence (AAAI-16)*. https://ojs.aaai.org/index.php/AAAI/article/download/9973/9832.

Jiang, Y., Xia, L., Wei, W., Luo, D., Lin, K., & Huang, C. (2024). DIFFMM: Multi-modal diffusion model for recommendation. In *Proceedings of the 32nd ACM international conference on multimediaNew York, NY, USA* (pp. 7591–7599). Association for Computing Machinery. https://doi.org/10.1145/3664647.3681498

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. Cs https://doi.org/10.48550/arXiv.1412.6980

Klema, V., & Laub, A. (1980). The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, *25*, 164–176. https://doi.org/10.1109/TAC.1980.1102314

Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., & Farhadi, A. (2022). Matryoshka representation learning. In *Proceedings of the 36th international conference on neural information processing SystemsRed Hook, NY, USA*. Curran Associates Inc. https://doi.org/10.5555/3600270.3602462

Li, G., Zhuo, J., Li, C., Hua, J., Yuan, T., Niu, Z., Ji, D., Wu, R., & Zhang, H. (2021). Multimodal visual adversarial Bayesian personalized ranking model for recommendation. *Information Sciences*, *572*, 378–403. https://doi.org/10.1016/j.ins.2021.05.022

Li, X., & She, J. (2017). Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, ACM, Halifax NS Canada* (pp. 305–314). https://doi.org/10.1145/3097983.3098077

Li, Y. C., & Zhan, J. (2023). Effect of dimensionality reduction on uncertainty quantification in trustworthy machine learning. In *2023 International conference on machine learning and cybernetics (ICMLC)* (pp. 326–332). https://doi.org/10.1109/ICMLC58545.2023.10327953

Liang, S., Pan, Z., Liu, W., Yin, J., & Rijke, M. D. (2024). A survey on variational autoencoders in recommender systems. *ACM Computing Surveys*, *56*. https://doi.org/10.1145/3663364

Liu, J., Capurro, D., Nguyen, A., & Verspoor, K. (2023). Attention-based multimodal fusion with contrast for robust clinical prediction in the face of missing modalities. *Journal of Biomedical Informatics*, *145*. https://doi.org/10.1016/j.jbi.2023.104466

Liu, Q., Hu, J., Xiao, Y., Zhao, X., Gao, J., Wang, W., Li, Q., & Tang, J. (2024). Multimodal recommender systems: A survey. *ACM Computing Surveys*, *57*. https://doi.org/10.1145/3695461

Luo, X., Wang, H., Wu, D., Chen, C., Deng, M., Huang, J., & Hua, X. S. (2023). A survey on deep hashing methods. *ACM Transactions on Knowledge Discovery from Data*, *17*, 1–50. https://dl.acm.org/doi/10.1145/3532624.

Malitesta, D., Cornacchia, G., Pomo, C., Merra, F. A., Noia, T. D., & Sciascio, E. D. (2024). Formalizing multimedia recommendation through multimodal deep learning. *ACM Transactions on Recommender Systems* https://doi.org/10.1145/3662738.

Mccallum, M. C., Korzeniowski, F., Oramas, S., Gouyon, F., Ehmann, A. F., Rao, P., Murthy, H. A., Srinivasamurthy, A., Bittner, R. M., Repetto, R. C., Goto, M., & Serra, X. (2022). Supervised and unsupervised learning of audio representations for music understanding. In M. Miron (Ed.), *Proceedings of the 23rd international society for music information retrieval conference, ISMIR 2022, Bengaluru, India* (pp. 256–263). https://archives.ismir.net/ismir2022/paper/000030.pdf.

Mcinnes, L., Healy, J., Saul, N., & Großberger, L. (2018). UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, *3*, 861. https://doi.org/10.21105/joss.00861

Mortensen, A. K. (2024). Anderskm/gputil. original-date: 2017-01-16T11:57. https://github.com/anderskm/gputil.

Nagrani, A., Yang, S., Arnab, A., Jansen, A., Schmid, C., & Sun, C. (2021). Attention bottlenecks for multimodal fusion. In *Proceedings of the 35th conference on neural information processing systems (NeurIPS 2021)*. F564ebbc35b1014ac498fafadd0-Paper.pdf https://proceedings.neurips.cc/paper/2021/file/76ba.

Oramas, S., Nieto, O., Barbieri, F., & Serra, X. (2017a). Multi-label music genre classification from audio, text and images using deep features. In S. J. Cunningham, Z. Duan, X. Hu, & D. Turnbull (Eds.), *Proceedings of the 18th international society for music information retrieval conference Suzhou, China* (pp. 23–30). https://ismir2017.smcnus.org/wp-content/uploads/2017/10/126_Paper.pdf.

Oramas, S., Nieto, O., Sordo, M., & Serra, X. (2017b). A deep multimodal approach for cold-start music recommendation. In *Proceedings of the 2nd workshop on deep learning for recommender systems Como Italy* (pp. 32–37). ACM. https://doi.org/10.1145/3125486.3125492

Park, C., Kim, D., Oh, J., & Yu, H. (2017). Do "also-viewed" products help user rating prediction? In *the 26th International conference on world wide web, international world wide web conferences steering committee Perth Australia* (pp. 1113–1122). https://doi.org/10.1145/3038912.3052581

Pingali, S. (2023). Enhanced movielens 100K. https://www.kaggle.com/datasets/lamarockzz/enhanced-movielens-100k.

Reddy, M. S., & Adilakshmi, T. (2014). Music recommendation system based on matrix factorization technique-SVD. In *2014 International conference on computer communication and informatics* (pp. 1–6). https://doi.org/10.1109/ICCCI.2014.6921744

Tian, Z., Pan, L., Yin, P., & Wang, R. (2021). Information fusion-based deep neural attentive matrix factorization recommendation. *Algorithms*, *14*, 281. https://doi.org/10.3390/a14100281

Torres, N. (2023). A multimodal user-adaptive recommender system. *Electronics*, *12*. https://www.mdpi.com/2079-9292/12/17/3709. https://doi.org/10.3390/electronics12173709

Truong, Q. T., Salah, A., & Lauw, H. (2021a). Multi-modal recommender systems: Hands-on exploration. In *Fifteenth ACM conference on recommender systems* (pp. 834–837).

Truong, Q. T., Salah, A., & Lauw, H. W. (2021b). Bilateral variational autoencoder for collaborative filtering. In *Proceedings of the 14th ACM international conference on web search and data mining, ACM, virtual event Israel* (pp. 292–300). https://doi.org/10.1145/3437963.3441759

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. U., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/hash/.

Řehůřek, R. (2023). Doc2Vec model - gensim. https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.HTML.

Wang, H., Wang, N., & Yeung, D. Y. (2015). Collaborative deep learning for recommender systems. In *ACM SIGKDD international conference on knowledge discovery and data mining*, New York, NY, USA (pp. 1235–1244). Association for Computing Machinery. Proceedings of the 21th. https://doi.org/10.1145/2783258.2783273

Wang, Q., Wei, Y., Yin, J., Wu, J., Song, X., & Nie, L. (2023a). DualGNN: Dual graph neural network for multimedia recommendation. *IEEE Transactions on Multimedia*, *25*, 1074–1084. https://doi.org/10.1109/TMM.2021.3138298

Wang, W., Duan, L. Y., Jiang, H., Jing, P., Song, X., & Nie, L. (2021). Market2Dish: Health-aware food recommendation. *ACM transactions on multimedia computing, communications, and applications*, *17*, 1–19. https://doi.org/10.1145/3418211

Wang, W., Xu, Y., Feng, F., Lin, X., He, X., & Chua, T. S. (2023). Diffusion recommender model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information RetrievalNew York, NY, USA* (pp. 832–841). Association for Computing Machinery. https://doi.org/10.1145/3539618.3591663

Wei, Y., Wang, X., Nie, L., He, X., Hong, R., & Chua, T. S. (2019a). MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM international conference on multimedia, ACM, Nice France* (pp. 1437–1445). https://doi.org/10.1145/3343031.3351034

Wei, Y., Wang, X., Nie, L., He, X., Hong, R., & Chua, T. S. (2019b). MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM international conference on multimedia New York, NY, USA* (pp. 1437–1445). Association for Computing Machinery. https://doi.org/10.1145/3343031.3351034

Yuan, M., Zhuang, F., Zhang, Z., Wang, D., & Dong, J. (2023). Knowledge-based multiple adaptive spaces fusion for recommendation. In *Proceedings of the 17th ACM conference on recommender systems, ACM Singapore* (pp. 565–575). https://doi.org/10.1145/3604915.3608787

Zarzour, H., Al-Sharif, Z., Al-Ayyoub, M., & Jararweh, Y. (2018). A new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques. In *2018 9th International conference on information and communication systems (icics)* (pp. 102–106). https://doi.org/10.1109/IACS.2018.8355449

Zhang, A., Yu, Y., Zhang, L., Gao, R., & Yin, H. (2025). Contrastive translation with dynamical temperature for sequential recommendation. *IEEE transactions on systems, man, and cybernetics: Systems*, *55*, 4273–4285. https://doi.org/10.1109/TSMC.2025.3550701

Zhang, G., Liu, Y., & Jin, X. (2020). A survey of autoencoder-based recommender systems. *Frontiers of Computer Science*, *14*, 430–450. https://doi.org/10.1007/s11704-018-8052-6

Zhang, J., Zhu, Y., Liu, Q., Wu, S., Wang, S., & Wang, L. (2021). Mining latent structures for multimedia recommendation. In *Proceedings of the 29th ACM international*

conference on multimedia New York, NY, USA (pp. 3872–3880). Association for Computing Machinery. https://doi.org/10.1145/3474085.3475259

Zhou, J. P., Cheng, Z., Perez, F., & Volkovs, M. (2020). TAFA: Two-headed attention fused autoencoder for context-aware recommendations. In *Fourteenth ACM conference on recommender systems, ACM, virtual event Brazil* (pp. 338–347). https://doi.org/10.1145/3383313.3412268

Zhou, X., He, J., Huang, G., & Zhang, Y. (2015). SVD-based incremental approaches for recommender systems. *Journal of Computer and System Sciences*, *81*, 717–733. https://doi.org/10.1016/j.jcss.2014.11.016

Zhou, X., Lin, D., Liu, Y., & Miao, C. (2023). Layer-refined graph convolutional networks for recommendation. In *2023 IEEE 39th International conference on data engineering (ICDE)* Los Alamitos, CA, USA (pp. 1247–1259). IEEE Computer Society. https://doi.org/10.1109/ICDE55515.2023.00100

Zhou, X., & Shen, Z. (2023). A tale of two graphs: Freezing and denoising graph structures for multimodal recommendation. In *Proceedings of the 31st ACM international conference on multimedia*, the 31st ACM international conference on multimedia New York, NY, USA (pp. 935–943). Association for Computing Machinery. https://doi.org/10.1145/3581783.3611943