AutoMARS: Searching to Compress Multi-Modality Recommendation Systems

Duc Hoang
University of Texas at Austin
Austin, United States
hoangduc@utexas.edu

Haotao Wang
University of Texas at Austin
Austin, United States
htwang@utexas.edu

Handong Zhao
Adobe Research
San Jose, United States
hazhao@adobe.com

Ryan Rossi
Adobe Research
San Jose, United States
ryrossi@adobe.com

Sungchul Kim
Adobe Research
San Jose, United States
sukim@adobe.com

Kanak Mahadik
Adobe Research
San Jose, United States
mahadik@adobe.com

Zhangyang Wang
University of Texas at Austin
Austin, United States
atlaswang@utexas.edu

ABSTRACT

Web applications utilize Recommendation Systems (RS) to address the problem of consumer over-choices. Recent works have taken advantage of multi-modality or multi-view[1], input information (such as user interaction, images, texts, rating scores) to boost recommendation system performance compared with using single-modality information. However, the use of multi-modality input demands much higher computational cost and storage capacity. On the other hand, the real-world RS services usually have strict budgets on both time and space for a good customer experience. As a result, the model efficiency of multi-modality recommendation systems has gained increasing importance. While unfortunately, to the best of our knowledge, there is no existing study of a generic compression framework for multi-modality RS. In this paper, we investigate, for the first time, how to compress a multi-modality recommendation system with a fixed budget. Assuming that input information from different modalities are of unequal importance, a good compression algorithm should learn to automatically allocate different resource budgets to each input, based on their importance in maximally preserving recommendation efficacy. To this end, we leverage the tools of neural architecture search (NAS) and distillation, and propose *Auto Multi-modAlity Recommendation System* (**AutoMARS**), a unified modality-aware model compression framework dedicated to multi-modality recommendation systems. We demonstrate the effectiveness and generality of **AutoMARS** by testing it on three different Amazon datasets of various sparsity. **AutoMARS** demonstrates superior multi-modality compression performance than previous state-of-the-art compression methods. For example on the Amazon Beauty dataset, we achieve on average a 20% higher accuracy over previous state-of-the-art methods, while enjoying 65% reduction over baselines.

[1]Terms 'modality' and 'view' are interchangeably used throughout the paper.

CCS CONCEPTS

• **Computing methodologies** → **Discrete space search**; **Continuous space search**; • **Information systems** → **Information retrieval**.

KEYWORDS

AutoML, Recommendation System, Multi Modality

ACM Reference Format:
Duc Hoang, Haotao Wang, Handong Zhao, Ryan Rossi, Sungchul Kim, Kanak Mahadik, and Zhangyang Wang. 2022. AutoMARS: Searching to Compress Multi-Modality Recommendation Systems. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557242

1 INTRODUCTION

Over the years, Recommendation System (RS) endures as the *de facto* front-facing solution for consumer over-choices, with Collaborative Filtering (CF) and Matrix Factorization (MF) [15, 24] remaining popular solutions. Ideally, RS learns user/item profiles via textual reviews or visual images to predict unseen interactions. In reality, actual engagements are rare [14], and most solutions have traditionally suffered learning users' preferences because of interaction sparsity and cold-start [1, 36].

Recently, research directions [29, 49] have found far better success focusing on utilizing multi-modality input to better capture different aspects of user preferences. For example, [49] shows ways to combine heterogeneous modalities in an end-to-end manner for top-N recommendations. On the other hand, [29] introduces a hand-crafted solution to combine flat and hierarchical side information to traditional user/item embeddings to boost recommendation accuracy. A few [27] simply graft side-information like numerical ratings to the end of the embeddings vectors for inclusion.

Not surprisingly, an increasing number of features correlates to rising demand for data storage and computation speed. At the scale of today's industry with its billions of users and items, heterogeneity places a significant burden onto the service distributors and disrupts user experience. Therefore, scalability and efficiency are at the forefront of desirable characteristics for a recommendation system. This begs a practical and straightforward research question: *How do we solve the crucial conflict between the practical use of multi-modality inputs for performance boosting and the requirements on efficiency of recommendation models at scale?*

One solution is to apply tried-and-true techniques typical for Deep Neural Network (DNN) such as knowledge distillation [12, 19], channel pruning[4, 11], or low-rank factorization [6, 15, 24, 50]. Others involve designing dedicated feature-utilization strategies like those for homogeneous embedding space found in [26, 35, 40, 41]. However, these methods often rely on empirical rules to allocate computational budgets among multiple modalities, and thus can lead to sub-optimal trade-offs between model efficiency and accuracy. In view of that, we aim to develop a general compression framework that *automatically* learns to optimally allocate the available computational budgets among modalities while maximally preserving model efficacy.

We therefore propose the first model compression method for multi-modality recommendation systems, termed **Auto M**ulti-mod**A**lity **R**ecommendation **S**ystem (**AutoMARS**). Unlike previous RS compression methods for single modality input data, **AutoMARS** is designed to be modality-aware. It utilizes Neural Architecture Search (NAS) to allocate computational budgets for each input modality automatically. AutoMars optimizes for two very important tasks, *resource allocation* and *modality fusion*, in a differentiable manner. It is therefore the first end-to-end solution dedicated to compressing a multi-modality recommendation system. Additionally, it largely outperforms existing state-of-the-art model compression methods developed for relevant purposes on three popular recommendation datasets, showing the necessity and effectiveness of our customized framework. In summary, our contributions are:

- We study a novel problem of model compression under a given budget constraint in the multi-modality recommendation system.
- We propose a generic modality-aware model compression framework **AutoMARS**. We optimize our search tasks differentially to find a compact model's size with unique strategy for fusion.
- We achieved on average a 65% reduction in size and a 20% improvement in performance over previous state-of-the-art methods on the Amazon Datasets.

2 PRIOR WORKS

2.1 Recommendation Systems

The continuous evolution of Recommendation Systems has developed different strategies to best supply users with the most relevant products [2, 10, 16, 17, 33, 37, 39, 46, 47]. Most notable works are: Hidasi et al. [18] is the first to propose using Recurrent Neural Network to capture long-session data to supplant Matrix Factorization based approaches. Covington et al. [8] proposed an algorithm for recommending YouTube videos by utilizing candidate generation model and a separate ranking model, thus showing the effectiveness of deep neural network in improving accuracy. Wu et al. [45] used Graph Neural Network to capture complex transition between items, thus achieving more accurate item embeddings. Recently, Huang et al. [21] proposed a multi-attention head model in recommendation to mitigate low efficiency problem in group recommendation. For a complete literature review on the general recommendation system topic, we refer the readers to these comprehensive surveys and books [2, 39, 42, 47].

Multi-modality RS. Nowadays, a large amount of multi-modality (e.g., text, images, audio) data is generated online and utilized by recommendation systems. Zhang et al. [49] proposed a more general approach with the Joint Representation Learning framework (JRL), where parallel networks are assigned to each input modality to generate modality-specific embeddings for users/items. Liu et al. [29] explored a novel framework to take advantage of flat and hierarchical side-information to improve recommendation performance. While using multi-modality data largely boosts recommendation accuracy, it induces undesirable inference-time overhead. In the sequential recommendation setting, there are also a few emerging works [27, 48]. In this work, we focus on the classic, non-sequential recommendation setting.

2.2 Model Compression

Model compression is the attempt to miniaturize existing frameworks while maintaining or improving performance. Knowledge Distillation (KD) [19] is a classic compression method. A small student model can achieve comparable performance to its larger counterpart by learning to fit soft labels. Pruning [13, 14, 43] finds sparse structure within a dense model in an attempt to reduce the model's weights and is a popular approach for real-world applications. Quantization [7, 13] reduces the model parameter and activation bit-width for model compression and acceleration. Neural architecture search (NAS) has also been applied to search for efficient model structures [38, 44].

2.3 Model Compression on Recommendation Systems

Recent interests in on-device recommendation processing have brought about the need for a small, lightweight recommendation system for resource-constraint platforms. LightRec[26] composes of "codebooks" (i.e look-up-tables) to store learned embedding vectors. The inference is performed by having products retrieving a unique combination of codebooks learned during training, where each user's preferences are encoded to every vector. Thus compatibility between user-product is whether the product's combination is of user's preference. Shi et al. [35] reduces the size of embeddings by exploiting complementary partition through the use of their proposed *quotient-remainder* trick. This trick works by using both the quotient and remainder function to produce two different and smaller embeddings combined to produce a unique compositional embedding. LLRec [41] or Light Location Recommender System

Table 1: Notations and definitions for common variables.

Name	Definition
k	Number of modalities/views
u, p	user/product embeddings
U, P	total users and products
m_i^u, m_i^p	the user/product embedding masks for the i-th view
D	Total embedding columns
ω	model's weights
$g(.)$	the ranking loss function
$f(.)$	the combinatory function
α_i	compression search space for the i-th view $\alpha_i = \{d_i^0, d_i^1, ...d_i^{\lvert\alpha_i\rvert}\}$
d_i	the total spatial allocation for the i-th view $d_i = \lvert m_i^u \rvert + \lvert m_i^p \rvert$
λ_i	searched scaling hyperparameter
CR	Compression ratio $\frac{d_i^j}{D}$

reduces the large embedding size using tensor-train decomposition. Mao et al. [31] proposed feature selection by first ranking the embedding columns base on Frobenius-norm and select the top-k columns. UMEC [34] jointly compresses input feature and network structure. It formulates the problem as a resource-constrained optimization problem and solves it by the ADMM algorithm. Other work compresses recommendation systems by quantizing the embedding matrix [3, 22].

In recent years, Neural Architectural Search (NAS) has been applied to compress recommendation systems. Zhao et al. [51] proposed AutoEmb, a differentiable NAS system to select a combination of user/item embedding sizes base on popularity defined as the number of interactions between user-item. Chen et al. [5] proposed an evolution NAS approach termed RULE. RULE dynamically selects blocks of embeddings from a large embedding vector to accommodate for different hardware constraints.

AutoMars uniquely compresses recommendation with multi modality in a collaborative manner. In contrast, prior works such as [26, 35, 41] are more specific and would require multiple learning stages for each assigned modality. On the other hand, side-information recommenders like [29] only use low-dimensional heterogeneous dimensions already provided in the dataset. Finally, quantization techniques such as [3, 22] can directly extend to our or any pruning works and are not a direct benchmark since they could be applied to any embedding space.

3 METHODOLOGY

In this section, we describe **AutoMARS**'s overall methodology for automatic heterogeneous features compression and fusion. We first introduce the framework notations and overall objective. Next, we explain how our contributions, multi-modality fusion search and compression search, can be applied generally for MF-based recommenders. Finally, we describe the optimization and fine-tuning process. Our framework is illustrated with Figure 1.

3.1 Model Search Space

3.1.1 Preliminary. We introduce our notations in detail here, and summarize them in Table 1. Let V_i denote the i-th modality (also known as the i-th view) in our heterogeneous space. Expressly, we set V_0 for text and V_1 for images. We describe each view V_i as a set containing $\{u_i, p_i, \omega_i, \theta_i\}$. Herein, we define the user and product embeddings $u_i \in \mathbb{R}^{U\times D}$ and $p_i \in \mathbb{R}^{P\times D}$ as learned embedding from some objective loss function L_i, where D is the embedding size, U is total users and P is total products or items. In addition, as part of V_i, we also define ω_i and θ_i as the learnable weights, and set of parameters necessary to facilitate u_i and p_i. With heterogeneity, we also formulate our multi-views set for users as u and product embeddings as p. Thus for a k-views top-N recommendation system, **AutoMARS** optimizes the following general loss function:

$$\underset{\forall k:\{\omega_k,\theta_k\}}{\arg\min}\ \mathbf{L}_{obj} = \sum_{(u,p^+)\in\mathbb{R}} g(f(u), f(p^+), f(p^-)) + \sum_{k} \lambda_k \mathbf{L}_k(\theta_k) \tag{1}$$

Where $f(.)$ denotes some multi-views combinatory function for set u and p, (u, p^+) refers to positive pair or actual interactions, i.e purchase, between a user to an item, while (u, p^-) denotes a lack of interaction. $g(.)$ is our ranking loss function. λ_k is a scaling-factor for each modality. Ultimately, the end goal is to produce $\hat{s}$ the predicted interaction matrix where $\hat{s} = u^T \cdot p$.

3.1.2 Multi-Modality Fusion Search. Prior works [29, 49] have illustrated the non-triviality of selecting fusion function and the importance it has on performance, especially from heterogeneous sources. Naturally we see an opportunity to surpass what is essentially feature selection methods using gradient signals, by facilitating an architecture search using NAS on the multi-views combinatory function, $f(.)$, for both u and p. Thus, let e_i be interchangeable for both u_i and p_i and E for U and P, we formally define our $f(.)$ search space as:

- Concat: $e = e_1 || e_2 || ... || e_k$, where $e \in \mathbb{R}^{E\times k * D}$.
- Sum: $e = e_1 + e_2 ... + e_k$, where $e \in \mathbb{R}^{E\times D}$
- Max: $e = max(stack([e_1, e_2, ..e_k]))$, where we select maximum scalar value for each feature columns.
- Mean: $e = \frac{1}{k}(e_1 + e_2 + ... + e_k)$

We shall refer to our fusion search space as β from now on.

3.1.3 Compression Search. Additionally, we automate features budget using the same differential search. Let $\bar{u_i} = u_i \bigodot m_i^u$ and , likewise, $\bar{p_i} = p_i \bigodot m_i^p$, where $m_i^u, m_i^p \in \{0,1\}^{U\times D}$ are the binary mask matrices derive from our compression search. Then $\bar{u} = \{\bar{u_1}\bar{u_2}...\}$ and $\bar{p} = \{\bar{p_1}\bar{p_2}...\}$ denote our compressed multi-views user and product embeddings sets. Armed with these notations, we further divide compression search into two sub-tasks: *resource allocation* and *feature selection*. During our searching stage, we adopt the same hard Gumbel-Softmax trick [23] to differentially optimize all categorical variables.

Resource allocation. This sub-task determines d_i, the total resource space allocated for the i-th view. Formally we denote $d_i = |m_i^u| + |m_i^p|$ and constrained it generally between 0 and $d_i < kD$. Let α be the architecture parameter representing d for all k-views. For each view, we denote that search space α_i as $\{d_i^j : 0 \leq j \leq |\alpha_i|\}$,

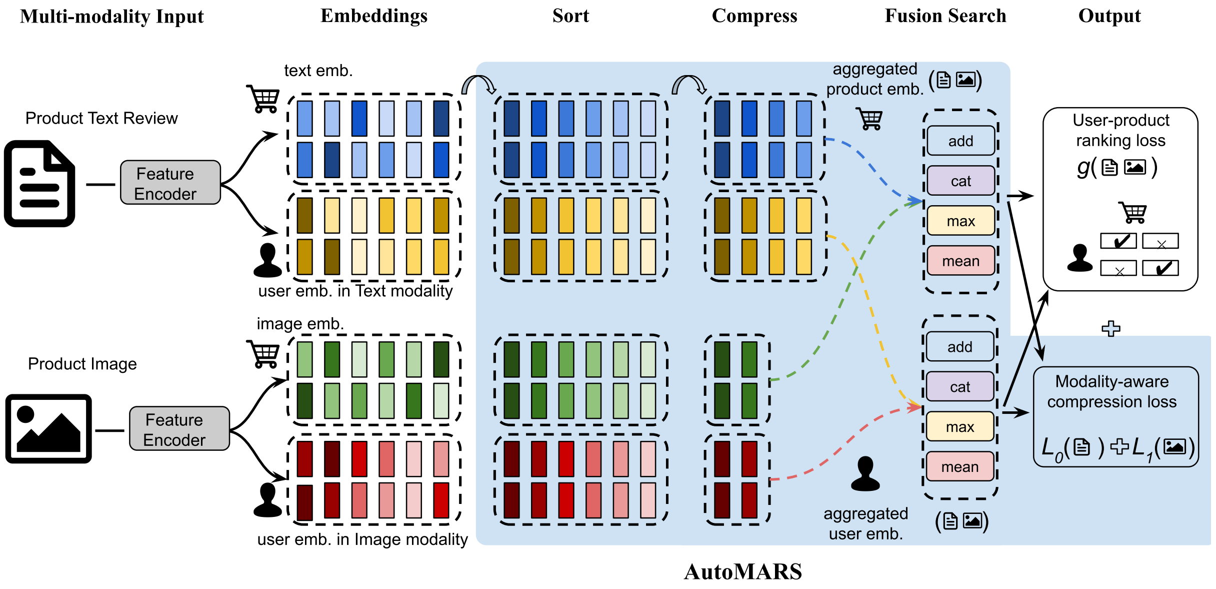


Figure 1: Illustration of our searching and compression process. AutoMARS is highlighted in blue shadow. On the left side, multi-modality inputs are processed via feature encoders into individual modality user/item embeddings before they are ranked, selected and fused. We automated both compression and fusion process to best maximize performance. The resulting compressed mix-modality user/item vectors are then judged on their performance and the whole process begins anew.

where d_i^j represents a choice for total space allotment of the i-th view. For k-views, our total search space $|\alpha|$ is D^k. In addition, to prevent unconstrained growth in compression search, we set α to be within the desired compression ratio (CR) range. CR is formally denote as $\frac{d_i^j}{D}$. By pre-defining the max and min of desired compression size, we limit the total possible states α can have. Thus, we denote our constrained search space, $\bar{\alpha}$, as:

$$\bar{\alpha} = \{\alpha_i \in \alpha | CR_{min} \leq \frac{d_i^j}{D} \leq CR_{max}, \forall j : 0, 1..|\alpha_i|\} \tag{2}$$

Henceforth, any reference to α is referring to $\bar{\alpha}$ for simplicity.

Feature Selection. While resource allocation controls the feature quantity, feature selection controls feature quality. We are inspired from Mao et al. [31] when implementing our own method. To illustrate our selection process, we will walk through the process for a single modality. Assume we obtained d_0^0, u_0, and p_0 for the text modality, the first step in our feature selection process is to normalize the embedding vectors across all users.

- ***Feature Normalization*** We perform $\mathbb{L}_2$-norm on both u_0 and p_0, so that $\mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{1\times D}$ and $\mathbb{R}^{P\times D} \rightarrow \mathbb{R}^{1\times D}$ respectively.
- ***Feature ranking*** After normalization, features are ranked base on their weights after we perform element wise multiplication $r_0 = ||u_0||_2 \bigodot ||p_0||_2$.
- **Feature selection** once r_0 is obtained, we perform $Top_k(r_0)$ where $k = d_0^0$ to obtains indexes of highest ranking feature pairs. After accomplishing this, we generate a binary masking vector, m_0^u and m_0^p to mask over u_0 and p_0.

We repeat the same process for all views.

3.2 Modality-aware Optimization

3.2.1 Modality objectives. Expressly, **AutoMARS** utilizes text and images to learn embeddings of users and items. To obtain the u_0 and p_0 from text, we adopt the simple PV-DBOW framework [25] with negative sampling strategy [32]. Formally, we define the objective loss for text as:

$$\begin{aligned}\mathbf{L}_0(\omega, t_{up}) = &\sum_{w\in V}\sum_{(u,p)\in\mathbb{R}} f_{w,t_{up}} \log\sigma(\omega^T t_{up}) \\ &+ \sum_{w\in V}\sum_{(u,p)\in\mathbb{R}} f_{w,t_{up}} (n\mathbb{E}_{w_N\sim P_V} \log\sigma(-\omega_N^T t_{up})\end{aligned} \tag{3}$$

Here, $t_{up} \in \mathbf{R}^D$ denotes a review embeddings between a user-item interaction. V defines the total vocabulary. $f_{w,t_{up}}$ represents frequency of word-review pair. σ denotes sigmoid function. Finally, $\mathbb{E}_{w_N\sim P_V}[\log\sigma(-\omega_N^T t_{up})]$ defines expected value of $\log\sigma(-\omega_N^T t_{up})$ given the unigram noise distribution P_V. t_{up} are then connected to the related user, and items.

To obtain u_1 and p_1 from images, we follow [49] and define our objective loss as:

$$\mathbf{L}_1(\omega, b, x_{up}) = \sum_{(u,p)\in\mathbf{R}} (ELU(\omega x_{uv} + b) - y_{uv})^2 \tag{4}$$

Where b is the linear bias, x_{up} is the user-item learned image embedding, y_{uv} is ground-truth embedding given to us. With these individual objective losses and architecture parameters defined, we rewrite **AutoMARS** overall objective function, Eq.1, as:

$$\begin{aligned}\mathbf{L}_{obj}(\bar{u}, \bar{p}, f(.), \omega, b, \alpha, \beta) = &\sum_{(\bar{u},\bar{p}^+)\in\mathbb{R}} g(f(\bar{u}), f(\bar{p}^+), f(\bar{p}^-)) \\ &+ \lambda_0\mathbf{L}_0(\omega_0, t_{\bar{u}\bar{p}}) + \lambda_1\mathbf{L}_1(\omega_1, b, x_{\bar{u}\bar{p}})\end{aligned} \tag{5}$$

3.2.2 Bi-level optimization. We adopt the GDAS [9] differential search process closely. With **AutoMARS** objective loss function clearly defined in Eq.5, we formally denote our bi-level search process as:

$$\min_{\alpha,\beta} \mathbf{L}_{obj}^{valid}(\bar{u},\bar{p},f(.),\omega,b,\alpha,\beta) \tag{6}$$

$$\text{s.t.} \quad \bar{u},\bar{p},f(.) = \underset{\forall k:\{\omega_k,\theta_k\}}{\arg\min}\ \mathbf{L}_{obj}^{train}(\bar{u},\bar{p},f(.),\omega,b,\alpha,\beta) \tag{7}$$

Note that α, β are optimized using objective loss function on the validation set, while u, p, ω, b are learned with training set. We adopt the same hard Gumbel-Softmax trick [23] as GDAS [9] to differentially optimize all categorical variables. Algorithm 1 illustrates our searching pseudo-code.

3.3 Fine-tuning and Self-distillation

3.3.1 Fine-tuning. Once the searching process has been completed, we use *argmax* function on α, β to obtain the final categorical choice. Furthermore, we use the selected sub-set of features obtained during the search from $\bar{u}, \bar{p}$ as our initial values for fine-tuning. This is because subset selection is part of the searching process as illustrated in figure 1, and we found doing so improves our performance empirically.

3.3.2 Self-distillation. Additionally, to further aid performance and consistency, we add $L_{distill}$ to equation 5 during fine-tuning. It is formulated as:

$$\mathbf{L}_{distill} = \sum_{i=0}^{k} D_{KL}(\bar{u}_i^T \bar{p}_i, u_i^{*T} p_i^*) \tag{8}$$

where D_{KL} is the Kullback-Leiber divergence. u_i^* and p_i^* are non-masked users and products embeddings found during the search for ith-view. In summary, $L_{distill}$ enables compressed models to learn a dense distribution of user-product interactions via self-distillation. As the name suggested, self-distillation uses already available information generated, i.e. u and p, as a mechanism to recover (and even improve) performance and reduce variation of $\bar{u}$ and $\bar{p}$.

4 EXPERIMENTS

This section highlights the extensive experiments and analysis comparing **AutoMars** against other compression works.

4.1 Dataset Description

The Amazon Review Dataset [14] provides a plethora of users' reviews, user-product interactions, as well as product's dense image features. It neatly categorizes all data into related domains, as denoted by the name of its sub-dataset in Table 2. Conveniently, it also provides 5-core dense datasets, a filtered dataset with user-product of at least five interactions. Since 5-core datasets are less noisy and resource-friendly, assume all datasets mentioned henceforth to be 5-core. Despite the relative density, 5-core datasets are still incredibly sparse, as we denote in table 2. We use four datasets with increasing density to show the effectiveness of our **AutoMARS**.

Algorithm 1: AutoMARS algorithm

```
Input: $\omega$, u, p, b, $\alpha$, $\beta$, data, epochs;
Output: $\bar{u}$, $\bar{p}$, $f(.)$;
Initialize $m^u, m^p$ as vector of ones;
for $e = 0$ to Epoch-1 do
    for $j = 0$ to $\|data\|$-1 do
        #sample masks and aggregation function from
        #architecture parameters with GumbelSoftmax
        $m^u, m^p, f(.) \leftarrow$ SampleArchitecture($\alpha, \beta$);
        #Generate mini-batch ($b$) for u and p on train data
        $u^b, p^b \leftarrow$ forward($\omega$, data$_j^{train}$, $u, p$);
        #Compute the masked embedding
        $\bar{u}^b \leftarrow \{u_i^b \odot m_i^u : 0 \le i \le k-1\}$;
        $\bar{p}^b \leftarrow \{p_i^b \odot m_i^p : 0 \le 1 \le k-1\}$;
        Backward $\mathbf{L}_{obj}^{train}(\bar{u}^b, \bar{p}^b, f(.), \omega, b, \alpha, \beta)$ update
         $\omega$,u,p;
        $u^b, p^b \leftarrow$ forward($\omega$, data$_j^{valid}$, $u, p$);
        $\bar{u}^b \leftarrow \{u_i^b \odot m_i^u : 0 \le i \le k-1\}$;
        $\bar{p}^b \leftarrow \{p_i^b \odot m_i^p : 0 \le 1 \le k-1\}$;
        Backward $\mathbf{L}_{obj}^{valid}(\bar{u}^b, \bar{p}^b, f(.), \omega, b, \alpha, \beta)$ update $\alpha, \beta$;
    end
end
#Arg Max to derive final settings
$m^u, m^p, f(.) \leftarrow$ DeriveArchitecture($\alpha, \beta$);
$\bar{u} \leftarrow \{u_i \odot m_i^u : 0 \le i \le k-1\}$;
$\bar{p} \leftarrow \{p_i \odot m_i^p : 0 \le 1 \le k-1\}$;
```

Table 2: Basic statistic of the five Amazon datasets

Dataset	#users	#products	#interactions	density
Movies	123,960	50,052	1,247,461	0.0201%
Home	66,519	28,237	551,682	0.0294%
Clothing	39,387	23,033	278,677	0.0307%
Cell Phones	27,879	10,429	194,439	0.0669%
Beauty	22,363	12,101	198,502	0.0734%

4.2 Experimental Setup

4.2.1 Baseline Methods. Since this is the first work to compress multi-modality recommendation systems, there are no previous baseline methods to compare on this task. Therefore we select representative compression methods from other related fields as our baseline methods.

- Unified Model and Embedding Compression (UMEC) [34]: UMEC jointly compresses input feature and network structure. It formulates the problem as a resource-constrained optimization problem and solves it by the ADMM algorithm. As a result, UMEC achieves state-of-the-art model compression performance on single-modality recommendation systems.
- Low-rank Multimodal Fusion (LMF) [30]: LMF Fusion is a popular feature compression method for multi-modality data by integrating multiple uni-modality representations into one compact multi-modality representation using low-rank

tensors. Unlike our method that dynamically learns the embedding size for each modality, LMF sets the embedding size as a hyper-parameter, where $d_i = D * \frac{CR}{2}$ is set as suggested in [30].

- Sparse Structure Selection (SSS) [20]: SSS is a widely-used channel pruning method for Convolutional Neural Networks (CNNs). It adds ℓ_1 sparsity constraints on channel-wise scaling factors and solves the optimization problem by a modified Accelerated Proximal Gradient (APG). SSS can be directly applied to any MF-based framework to compress the embedding matrix by adding column-wise scaling factors on the embedding matrix.

4.2.2 Implementation Details. All compression techniques, ours included, are searched for 20 epochs and trained for 50 epochs. The only exception being Movies with only 10 epochs due to time-constraint. To keep things fair, we maintain identical settings during embedding generation given a dataset. As for the learning rate, we use a cyclical learning rate adapted from Zhang et al. [49] implementation, for both the model and architecture's learning rates. We set the starting learning rate value to be 0.5 for model's and 0.001 for architecture's. Additionally, we set τ for $Gumbel-softmax$ [23] to be at ten and linearly decreasing after each epoch to a minimum value at 0.001. We use the SGD optimizer for the model's weights and the Adam optimizer for the model's architecture. By default, we use concatenation function for $f(.)$. Finally, we set our batch size for words to be 512.

The embedding learning scheme is partially-adapted from the methodology proposed by Zhang et al. [49]. We use it exclusively to conduct our multi-modality compression experiments. In all experiments, we set the maximum number of features, D, to be 300 features. For text view, we set λ_0 to be 1 for all datasets. For image view, we set λ_1 to be 0.001 for Beauty , Cell Phones, Home and Movies dataset, and 0.1 for Clothing dataset. Next, we conduct study on the robustness of our method by searching for three separate sets of min-max CR settings. Given k-views and D-dimension, our min-max pair for CR for each settings are $(0.8kD - 0.6kD)$, $(0.6kD - 0.4kD)$ and $(0.4kD - 0.2kD)$, where $k = 2$ and $D = 300$. These settings are denoted as "Large", "Medium", "Small" respectively on Table 3, 4 and 5.

4.2.3 Evaluation Metrics. To evaluate performance, we use the following metrics:

- NDCG: This measure of ranking quality that takes correctly recommended items' position into account averaged over all users.
- Recall: percentage of recommended item truly intended for a user list averaged across all users.
- Precision: The average percentage across all users of correctly recommended items.
- HT: Hit Ratio measures the percentage of users with at least one correctly recommended item in their list.
- KByte: the measurement of allocated space of total embedding in Kilo-Byte.
- GPU/CPU(ms): running time in milliseconds on NVIDIA 2080ti GPU and intel-i7 CPU.

We compare performance using the Top-N recommendation list for each user in the testing set. Here we set N = 10.

4.3 Main Results

We extensively compare our method against other compression works, and collect their performance and efficiency results in Table 3 and Table 4 respectively. From these results, we drew some consistent observations:

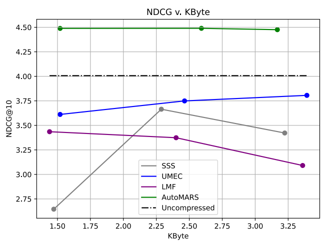
- About modality composition, experiments on full-size embeddings settings suggest an asymmetrical preference toward one form of representation. We observe minor preference in the Clothing and Beauty dataset toward the image (+0.07%) and the text(0.06%) modality. While Cell Phone and Home exhibit more significant preference with 0.64% and 0.96% differences between the two's modality performances. Such observations offer an incentive for a modality-aware compressor, such as ours, to find efficient but effective embedding composition.
- We do not always observe a positive correlation between size and performance when comparing different compression settings. Generally, **AutoMars**, SSS [20] and LMF [30] perform better with increasingly smaller embedding sizes. As an example, in comparison to the performance of our full-size benchmark on row one, **AutoMars** gains 0.28%, 0.30%, 0.29% in performance while shedding 1.63KB, 2.208KB, 3.280KB for the Cell Phone dataset, respectively. The same holds across our tested datasets regardless of their relative density. This increase in performance toward smaller embeddings coincides with our previous observation of the asymmetric preference between our modality and suggests compression behaves similarly to a filter by reducing noisy features from the nonpreferential modality without eradicating them.
- Comparing different compression methods efficiency, we observe a positive correlation between embedding size (Kbytes) and latency (GPU/GPU speed in milliseconds (ms)). Table 4 details our experiments with latency between our method and different baseline methods. Typically, **AutoMars** is a better compressor under Large and Medium compression ratio (with the exception for Home), while LMF [30] dominates the Small compression ratio due to its inherent inflexibility. However, when we correlate our performance with Table 3, **AutoMars** is much more effective at choosing the optimal size and features given the budget. Figure 2 illustrates our performance to size effectiveness of all datasets experimented. Finally, we observe some slight discrepancy between size and latency — in Large Clothing, **AutoMars**'s is $0.008Kbyte$ smaller than LMF's but is $0.121ms$ slower in GPU time. However, we deem this result acceptable since the time discrepancy is within our standard deviation and the differences between size is insignificant.
- Observing how the resource is allocated, we see that given a budget, **AutoMars** does not aim for the lowest possible embedding size. From a given search space, **AutoMars** seeks to balance budgets and performance. Since our constraints are fairly relaxed, perhaps with a more restrictive seach space, we can force **AutoMars** to generate a smaller embedding

Table 3: Summary of performance for AutoMARS and baselines. Results are presented in percentage with % omitted. Bold numbers indicate the best performing scores across all models for a particular metric. The first three rows illustrate performance of a full-size multi-modality and individual modality embedding performance. All measurements are @10

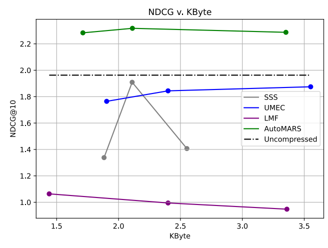
		Cell Phone				Clothing				Beauty				Home				Movie			
		NDCG	Recall	Prec	HT	NDCG	Recall	Prec	HT	NDCG	Recall	Prec	HT	NDCG	Recall	Prec	HT	NDCG	Recall	Prec	HT
	Image-text	4.19	7.47	1.10	10.91	1.92	3.52	0.50	5.26	4.96	8.07	1.68	14.01	1.46	2.47	0.44	4.37	0.17	0.26	0.10	0.97
	Text-only	4.00	7.17	1.05	10.42	1.94	3.51	0.49	5.22	4.75	7.59	1.60	13.25	1.17	1.98	0.36	3.57	0.20	0.29	0.10	0.98
	Image-only	3.36	5.68	0.85	8.35	2.01	3.51	0.50	5.24	4.69	6.91	1.55	12.36	0.21	0.33	0.09	0.89	**0.29**	**0.48**	**0.15**	**1.55**
Large	UMEC	3.81	6.80	1.00	9.84	1.87	3.46	0.49	5.12	4.81	7.75	1.65	13.53	0.40	0.69	0.14	1.42	0.19	0.25	0.10	1.01
	SSS	3.42	5.89	0.86	8.49	1.41	2.47	0.34	3.74	4.69	7.41	1.59	13.00	1.13	1.91	0.34	3.44	0.20	0.22	0.09	0.90
	LMF	3.09	5.45	0.78	7.91	0.95	1.61	0.23	2.48	3.31	4.97	1.13	9.01	0.05	0.09	0.02	0.25	0.03	0.04	0.01	0.12
	AutoMARS	**4.48**	**7.95**	**1.17**	**11.38**	**2.29**	**4.16**	**0.59**	**6.11**	**5.25**	**8.47**	**1.78**	**14.55**	**1.50**	**2.57**	**0.46**	**4.50**	0.21	0.29	0.11	1.13
Medium	UMEC	3.75	6.78	1.00	9.89	1.84	3.36	0.48	5.00	4.80	7.71	1.64	13.57	0.39	0.76	0.16	1.55	0.14	0.19	0.07	0.69
	SSS	3.66	6.23	0.90	8.86	1.91	3.47	0.49	5.10	4.29	7.04	1.49	12.27	1.09	1.84	0.34	3.33	0.15	0.20	0.08	0.80
	LMF	3.37	5.84	0.82	8.43	0.99	1.66	0.25	2.61	3.34	5.17	1.16	8.95	0.06	0.10	0.03	0.29	0.04	0.06	0.01	0.15
	AutoMARS	**4.49**	**7.99**	**1.17**	**11.55**	**2.32**	**4.15**	**0.59**	**6.11**	**5.23**	**8.39**	**1.75**	**14.45**	**1.46**	**2.47**	**0.45**	**4.37**	0.15	0.21	0.08	0.83
Small	UMEC	3.61	6.61	0.98	9.64	1.76	3.22	0.45	4.75	4.60	**7.48**	1.57	13.14	0.43	0.76	0.16	1.55	0.12	0.19	0.07	0.69
	SSS	2.65	4.82	0.73	7.25	1.33	2.27	0.33	3.42	**4.67**	7.32	1.57	**12.96**	1.11	1.86	0.33	3.33	0.13	0.16	0.07	0.77
	LMF	3.43	5.96	0.85	8.62	1.06	1.68	0.25	2.60	1.18	5.42	1.22	9.84	0.04	0.06	0.02	0.22	0.03	0.05	0.01	0.14
	AutoMARS	**4.49**	**7.97**	**1.17**	**11.51**	**2.28**	**4.09**	**0.58**	**6.04**	4.66	7.39	**1.59**	12.95	**1.44**	**2.44**	**0.44**	**4.30**	0.15	0.20	0.08	0.79

Table 4: Embedding size and prediction latency measurements of approaches reported in Table 3. We report the average latency value of 1000 iterations between 2000 products/users. By default, full-size embedding of both Image-Text is 4.8 KByte with run time of 3.950±0.224ms and 10.841±1.115ms on GPU/CPU. {Note that LMF reports the same result across datasets because of its inability of dynamically learning embedding size.}

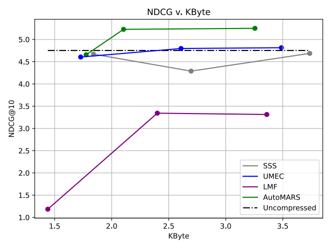
		Cell Phone			Clothing			Beauty			Home			Movie		
		KByte	GPU(ms)	CPU(ms)	KByte	GPU(ms)	CPU(ms)	KByte	GPU(ms)	CPU(ms)	KByte	GPU(ms)	CPU(ms)	KByte	GPU(ms)	CPU(ms)
Large	UMEC	3.39	2.81±0.19	7.97±0.61	3.55	3.09±0.21	7.82±0.60	3.49	2.98±0.18	7.81±0.84	3.82	3.52±0.27	8.02±0.73	2.97	2.97±0.24	8.86±1.12
	SSS	3.22	2.85±0.17	7.61±0.63	2.55	2.37±0.19	5.48±0.49	3.74	3.35±0.25	8.29±0.78	**3.02**	2.71±0.23	6.00±0.53	**2.34**	2.57±0.11	6.38±1.07
	LMF	3.36	2.91±0.21	7.70±0.60	3.36	2.91±0.21	7.70±0.60	3.36	2.91±0.21	7.70±0.60	3.36	2.91±0.21	7.70±0.60	3.36	2.91±0.21	7.70±0.60
	AutoMARS	**3.17**	2.74±0.21	7.49±0.56	**3.35**	3.03±0.21	7.54±0.58	**3.26**	2.80±0.15	7.68±0.60	3.79	3.40±0.23	8.04±0.70	3.55	3.84±0.45	14.54±4.52
Medium	UMEC	2.46	2.16±0.15	5.39±0.49	2.74	2.37±0.16	5.87±0.52	2.61	2.37±0.18	5.62±0.57	2.85	2.52±0.52	5.92±0.48	2.49	2.65±0.15	6.72±1.06
	SSS	**2.29**	2.15±0.16	5.05±0.38	2.40	2.11±0.22	5.42±0.43	2.70	2.47±0.22	5.99±0.55	**1.88**	1.72±0.18	3.65±0.36	**2.23**	2.55±0.16	6.19±1.24
	LMF	2.40	2.10±0.17	5.36±0.44	2.40	2.10±0.17	5.36±0.44	2.40	2.10±0.17	5.36±0.44	2.40	2.10±0.17	5.36±0.44	2.40	2.10±0.17	5.36±0.44
	AutoMARS	2.59	2.34±0.21	5.63±0.52	**2.11**	1.75±0.16	4.99±0.52	**2.10**	1.93±0.15	4.89±0.50	2.30	2.11±0.22	5.13±0.41	2.60	2.82±0.13	9.71±3.62
Small	UMEC	1.52	1.53±0.15	3.08±0.33	1.90	1.76±0.19	4.04±0.38	1.73	1.49±0.15	3.47±0.40	**0.51**	0.63±0.08	1.42±0.15	2.39	2.52±0.11	6.37±0.73
	SSS	1.47	1.30±0.11	3.08±0.30	2.14	1.88±0.18	4.80±0.49	1.84	1.70±0.14	3.78±0.41	1.86	1.58±0.16	3.66±0.55	1.67	1.82±0.13	4.22±0.85
	LMF	**1.44**	1.37±0.14	3.02±0.27	**1.44**	1.37±0.14	3.02±0.27	**1.44**	1.37±0.14	3.02±0.27	**1.44**	1.37±0.14	3.02±0.27	1.44	1.37±0.14	3.02±0.27
	AutoMARS	1.52	1.52±0.17	3.14±0.29	1.71	1.57±0.16	3.62±0.53	1.78	1.72±0.17	3.62±0.36	1.02	1.06±0.11	2.33±0.33	**1.22**	1.41±0.14	4.54 ±2.17



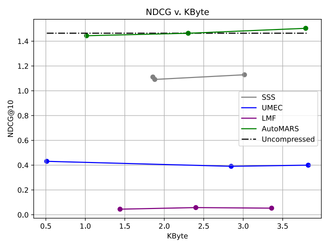
(a) Cell Phone



(b) Clothing



(c) Beauty



(d) Home

Figure 2: We illustrate the performance gain (or lost) at different size in Kbytes for each method. We marked the baseline's performance irrespective of its size as the horizontal dash line for perspective. AutoMars achieves consistently great performance across different variation of model's size.Observe that size and performance are not always positively correlated. The increase in performance toward smaller embeddings coincides with our previous observation on the asymmetric preference between our modality and suggests compression behaves similarly to a filter by reducing noisy features from the nonpreferential modality.

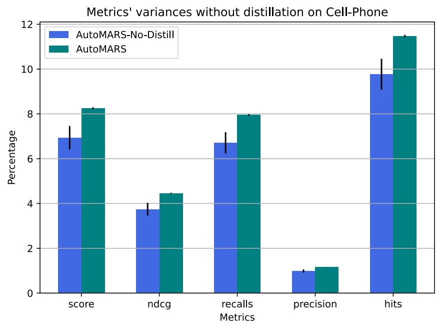


Figure 3: Stability analysis of AutoMARS with or without distillation. The error bars (mean ± standard derivation) are obtained over six random runs. Experiments are conducted on Cell-Phone and medium model size. Self-distillation helps stabilize AutoMars performance and improve accuracy.

size to the detriment of performance. Other methods are easily biased toward small settings since they can trivially reduce one-half of the constraint that way.

- Comparing different compression methods performance, **AutoMars** generally achieves much better performance — For example, under NDCG@10, **AutoMars** gains over the next best in Large compression setting at 0.67% for Cell-Phone, 0.41% in Clothing, 0.44% for Beauty, 0.38% for Home and 0.02% for Movies. Compared to the full-size embeddings benchmark, it manages to gain extra performance, 0.5% for Cell Phone, 1.4% for Clothing, 1.3% for Beauty, and 0.04% for Home. The only notable exception being Movies, where it did not gain any performance over baseline image. Lastly, we observe our performance gain against other compression frameworks does not clearly correlate with the dataset's interaction density, while against full-size embeddings, our improvement in performance correlates negatively with decreasing interaction ratio. Figure 2 illustrate said phenomenon clearly when compared to other compression methods and full-size embeddings.

4.4 Ablation Study

For this section, we analyze the merit of each of our components. These include self-distillation, compression ratio, features' identities and multi-modality fusion methods. To do this, we add three different variants of **AutoMARS** with some features missing.

- In Random-selection, we disregard the index positions of selected features and embedding values found during search. Instead, we keep only the modality preference and fine-tune it from scratch.
- In No-Distill, we fine-tune our network without distillation loss while keeping everything else identical.
- In Half-half, we equally represent each modality with the searched total number of features.

4.4.1 Impact of self-distillation. Inspired by the recent finding that model distillation can effectively stabilize model compression process [40], we integrate NAS with self-distillation. We observe that

Table 5: Comparison between different ablation studies to the baseline at identical compression ratios for Cell-Phone dataset.

		NDCG	Recall	Prec	HT
Large	half-half	3.9327	7.0695	1.0386	10.2658
	No-Distill	4.4409	7.8889	1.1576	11.3849
	Random-selection	3.5810	6.5512	0.9623	9.5054
	AutoMars	**4.4756**	**7.9476**	**1.1671**	**11.3849**
Medium	half-half	3.8914	7.0442	1.0347	10.2407
	No-Distill	4.1462	7.3792	1.0859	10.7464
	Random-selection	3.5333	6.3845	0.9372	9.2758
	AutoMARS	**4.4895**	**7.9952**	**1.1736**	**11.5463**
Small	half-half	3.7779	6.8482	1.0069	9.9788
	No-Distill	3.9438	7.1809	1.0506	10.4882
	Random-selection	3.8152	6.8693	1.006	9.9501
	AutoMARS	**4.4886**	**7.9694**	**1.1700**	**11.514**

this simple modification can improve performance and maintain consistency. We demonstrate how self-distillation stabilizes the fine-tuning process with figure 3. This figure illustrates the wide deviation across all metrics between using self-distillation and not using distillation. In addition, it shows self-distillation significantly reduce all standard deviation across all metrics. Furthermore, figure 4 shows that distillation improves performance for all model sizes, especially in small compression setting, where we observe a disproportional gain in performance, of up to 20% for Clothing. Table 5 clearly summarizes these observations.

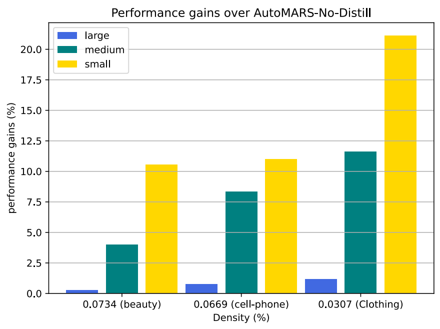


Figure 4: Performance percentage gains versus interaction density for all model sizes. Percentage gains are the percentage difference of recalls between AutoMARS and AutoMARS-No-Distill. As the dataset gets increasingly sparser, we observe larger gains using distillation, especially for the smaller model. This trend is consistence for all metrics.

4.4.2 Impact of Features Selection. In **AutoMARS**, we utilize the feature subsets found during the search as initial values for fine-tuning. We do this because feature sorting is part of the compression process, which would be destroyed during fine-tuning if we perform zero-initialization, nullifying the search result. Additionally, since we searched on embedding vectors, the derived α is naturally

Table 6: Comparison between different multi-views fusion strategies apply to uncompressed baseline using default configurations. Cell-Phone is used for this experiment.

	NDCG	Recall	Prec	HT
Concat	4.1903	7.4761	1.1041	10.9186
Max	4.2437	7.3935	1.0923	10.6854
Add	4.1639	7.2603	1.0568	10.3518
Mean	3.5598	6.3403	0.9479	9.3367

unstructured, which is unlike most NAS [28].Thus, to create a regular structure like network architecture, we place importance on positions and scalar values of for searched subset of features.

To validate this hypothesis, we test whether inheriting the index positions and scalar values for selected features during the search can help our fine-tuning process achieve higher accuracy than without the inheritance. We compare the results of our two variants Random and No-Distill, and report their performances under table 5 and figure 2. We can see that we consistently achieve much better results by inheriting the index and scalar values of the feature subset for fine-tuning. This result empirically supports our hypothesis and validates our approach for **AutoMARS**.

4.4.3 Impact of Multi-Modality Fusion. We study the impact of different fusion strategies on the baseline framework. The results are summarized in Table 6. Under the ideal circumstance, concatenation remains the ideal choice for multi-views fusion, followed by max, add, and mean. Intuitively, this makes sense, as concatenation allows for homogeneous scaling of the product's modality features between the user and product embeddings before the final summation of all features. Max and Add functions enforce features that can be similarly interpreted across different modalities into the same spatial location of the embedding vector. On the other hand, the Mean function normalizes features across modalities and smooths the embedding space, which can cause features to be overly smoothed and thus lower the overall embedding dimension.

4.4.4 Intuitions. From the abundant empirical results collected, we can derive a tuition to when to use which method base on our needs. For instance, for a general min-maxing of budget and performance, **AutoMARS** would be the best choice for its ability to maintain performance while keeping the budget low. However, due to the nature of Neural Architecture Search, we find having precise control over the exact budget more challenging. Therefore, in a scenario where budget is of the utmost priority, LMF is the better choice thanks to its predictable nature. On the other hand, SSS and UMEC are relatively simpler to implement. In the case where we want fast results, we recommend SSS or UMEC.

5 CONCLUSION

In this paper, we investigate, for the first time, how to compress a multi-modality recommendation system. We propose **AutoMARS** the first model compression method for multi-modality recommendation systems. Unlike previous RS compression methods for single modality input data, **AutoMARS** is designed to be modality-aware. It utilizes Neural Architecture Search (NAS) to allocate computational budgets for each input modality automatically. We show that having correct modality-preference can improve the model performance at a much smaller memory footprint. We further improve accuracy by recognizing the importance of features positions and identities. Additionally, we demonstrate that distillation stabilizes performance and improves gains for smaller models at sparser datasets. Compare to existing compression works, **AutoMARS** achieves on average a 20% increase in accuracy over baseline while enjoying a 65% reduction in overall size. A current limitation is the absence of experiments on edge devices due to time constraints. In future works, we will address this and further investigate more advanced and efficient modality fusion approaches. We want to experiment with block-wise dynamic feature selection based on user-product pairing across different modalities targeting performance using NAS.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (June 2005), 734–749. https://doi.org/10.1109/TKDE.2005.99 Copyright: Copyright 2011 Elsevier B.V., All rights reserved..

[2] Charu C. Aggarwal. 2016. *Recommender Systems - The Textbook.* Springer.

[3] Thalaiyasingam Ajanthan, Puneet K. Dokania, Richard Hartley, and Philip H. S. Torr. 2019. Proximal Mean-field for Neural Network Quantization. arXiv:1812.04353 [cs.CV]

[4] Tianlong Chen, Xuxi Chen, Xiaolong Ma, Yanzhi Wang, and Zhangyang Wang. 2022. Coarsening the Granularity: Towards Structurally Sparse Lottery Tickets. arXiv:2202.04736 [cs.LG]

[5] Tong Chen, Hongzhi Yin, Yujia Zheng, Zi Huang, Yang Wang, and Meng Wang. 2021. Learning Elastic Embeddings for Customizing On-Device Recommenders. arXiv:2106.02223 [cs.IR]

[6] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* (2017).

[7] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2016. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. arXiv:1511.00363 [cs.LG]

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

[9] Xuanyi Dong and Yi Yang. 2019. Searching for A Robust Neural Architecture in Four GPU Hours. arXiv:1910.04465 [cs.CV]

[10] Michael D. Ekstrand, John Riedl, and Joseph A. Konstan. 2011. Collaborative Filtering Recommender Systems. *Found. Trends Hum. Comput. Interact.* 4, 2 (2011), 175–243.

[11] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. arXiv:1803.03635 [cs.LG]

[12] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge Distillation: A Survey. *International Journal of Computer Vision* 129, 6 (Mar 2021), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z

[13] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv:1510.00149 [cs.CV]

[14] Ruining He and Julian McAuley. 2016. Ups and Downs. *Proceedings of the 25th International Conference on World Wide Web* (Apr 2016). https://doi.org/10.1145/2872427.2883037

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. arXiv:1708.05031 [cs.IR]

[16] Zhankui He, Handong Zhao, Zhe Lin, Zhaowen Wang, Ajinkya Kale, and Julian J. McAuley. 2021. Locker: Locally Constrained Self-Attentive Sequential Recommendation. In *The 30th ACM International Conference on Information and Knowledge Management.* 3088–3092.

[17] Zhankui He, Handong Zhao, Tong Yu, Sungchul Kim, Fan Du, and Julian J. McAuley. 2022. Bundle MCR: Towards Conversational Bundle Recommendation. *CoRR* abs/2207.12628 (2022).

[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939 [cs.LG]

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531 [stat.ML]

[20] Zehao Huang and Naiyan Wang. 2018. Data-Driven Sparse Structure Selection for Deep Neural Networks. arXiv:1707.01213 [cs.CV]

[21] Zhenhua Huang, Xin Xu, Honghao Zhu, and Meng Chu Zhou. 2020. An Efficient Group Recommendation Model with Multiattention-Based Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 31, 11 (Nov. 2020), 4461–4474. https://doi.org/10.1109/TNNLS.2019.2955567

[22] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. arXiv:1609.07061 [cs.NE]

[23] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. arXiv:1611.01144 [stat.ML]

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS.

[25] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. arXiv:1405.4053 [cs.CL]

[26] Defu Lian, Haoyu Wang, Zheng Liu, Jianxun Lian, Enhong Chen, and Xing Xie. 2020. Lightrec: A memory and search-efficient recommender system. In *Proceedings of The Web Conference 2020*. 695–705.

[27] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Non-invasive Self-attention for Side Information Fusion in Sequential Recommendation. arXiv:2103.03578 [cs.IR]

[28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. arXiv:1806.09055 [cs.LG]

[29] Tianqiao Liu, Zhiwei Wang, Jiliang Tang, Songfan Yang, Gale Yan Huang, and Zitao Liu. 2019. Recommender systems with heterogeneous side information. In *The World Wide Web Conference (WWW)*. 3027–3033.

[30] Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. 2018. Efficient Low-rank Multimodal Fusion with Modality-Specific Factors. arXiv:1806.00064 [cs.AI]

[31] Xueyu Mao, Saayan Mitra, and Viswanathan Swaminathan. 2017. Feature Selection for FM-Based Context-Aware Recommendation Systems. In *2017 IEEE International Symposium on Multimedia (ISM)*. 252–255. https://doi.org/10.1109/ISM.2017.42

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Neural and Information Processing System (NIPS)*. https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[33] J. Ben Schafer, Dan Frankowski, Jonathan L. Herlocker, and Shilad Sen. 2007. Collaborative Filtering Recommender Systems. In *The Adaptive Web, Methods and Strategies of Web Personalization (Lecture Notes in Computer Science, Vol. 4321)*, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Springer, 291–324.

[34] Jiayi Shen, Haotao Wang, Shupeng Gui, Jianchao Tan, Zhangyang Wang, and Ji Liu. 2021. UMEC: Unified model and embedding compression for efficient recommendation systems. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BM---bH_RSh

[35] Hao-Jun Michael Shi, Dheevatsa Mudigere, Maxim Naumov, and Jiyan Yang. 2020. Compositional embeddings using complementary partitions for memory-efficient recommendation systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 165–175.

[36] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Adv. in Artif. Intell.* 2009, Article 4 (Jan. 2009), 1 pages. https://doi.org/10.1155/2009/421425

[37] Karthik Subbian, Charu C. Aggarwal, and Kshiteesh Hegde. 2016. Recommendations For Streaming Data. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi (Eds.). ACM, 2185–2190.

[38] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2820–2828.

[39] Vipul Vekariya and G. R. Kulkarni. 2012. Hybrid recommender systems: Survey and experiments. In *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), Bangkok, Thailand, May 16-18, 2012*. IEEE, 469–473.

[40] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. 2020. GAN Slimming: All-in-One GAN Compression by A Unified Optimization Framework. In *European Conference on Computer Vision (ECCV)*. Springer, 54–73.

[41] Qinyong Wang, Hongzhi Yin, Tong Chen, Zi Huang, Hao Wang, Yanchang Zhao, and Nguyen Quoc Viet Hung. 2020. Next point-of-interest recommendation on resource-constrained mobile devices. In *Proceedings of the Web conference 2020*. 906–916.

[42] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. 2022. A Survey on Session-based Recommender Systems. *ACM Comput. Surv.* 54, 7 (2022), 154:1–154:38.

[43] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning Structured Sparsity in Deep Neural Networks. arXiv:1608.03665 [cs.NE]

[44] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10734–10742.

[45] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (Jul 2019), 346–353. https://doi.org/10.1609/aaai.v33i01.3301346

[46] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard de Melo, S. Muthukrishnan, and Yongfeng Zhang. 2020. CAFE: Coarse-to-Fine Neural Symbolic Reasoning for Explainable Recommendation. In *The 29th ACM International Conference on Information and Knowledge Management*. ACM, 1645–1654.

[47] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *CoRR* abs/1707.07435 (2017). http://arxiv.org/abs/1707.07435

[48] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation.. In *IJCAI*. 4320–4326.

[49] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1449–1458.

[50] Handong Zhao, Zhengming Ding, and Yun Fu. 2017. Multi-View Clustering via Deep Matrix Factorization. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 2921–2927.

[51] Xiangyu Zhao, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Jiliang Tang. 2020. AutoEmb: Automated Embedding Dimensionality Search in Streaming Recommendations. arXiv:2002.11252 [cs.IR]