Automated Machine Learning for Deep Recommender Systems: A Survey

Bo Chen\textsuperscript{1}, Xiangyu Zhao\textsuperscript{2}, Yejing Wang\textsuperscript{2}, Wenqi Fan\textsuperscript{3}, Huifeng Guo\textsuperscript{1}, Ruiming Tang\textsuperscript{1}

\textsuperscript{1}Huawei Noah's Ark Lab, \textsuperscript{2}City University of Hong Kong, \textsuperscript{3}The Hong Kong Polytechnic University

\{chenbo116, huifeng.guo, tangruiming\}@huawei.com, xianzhao@cityu.edu.hk, adave631@gmail.com, wenqifan03@gmail.com

Abstract

Deep recommender systems (DRS) are critical for current commercial online service providers, which address the issue of information overload by recommending items that are tailored to the user's interests and preferences. They have unprecedented feature representation effectiveness and the capability of modeling the non-linear relationships between users and items. Despite their advancements, DRS models, like other deep learning models, employ sophisticated neural network architectures and other vital components that are typically designed and tuned by human experts. This article will give a comprehensive summary of automated machine learning (AutoML) for developing DRS models. We first provide an overview of AutoML for DRS models and the related techniques. Then we discuss the state-of-the-art AutoML approaches that automate the feature selection, feature embeddings, feature interactions, and system design in DRS. Finally, we discuss appealing research directions and summarize the survey.

1 Introduction

Recent years have witnessed the explosive growth of online service providers [Ricci et al., 2011], including a range of scenarios like movies, music, news, short videos, e-commerces, etc. This leads to the increasingly serious information overload issue, overwhelming web users. Recommender systems are effective mechanisms that mitigate the above issue by intelligently retrieving and suggesting personalized items, e.g., contents and products, to users in their information-seeking endeavors, so as to match their interests and requirements better. With the development and prevalence of deep learning, deep recommender systems (DRS) have piqued interests from both academia and industrial communities [Zhang et al., 2019; Nguyen et al., 2017], due to their superior capacity of learning feature representations and modeling non-linear interactions between users and items [Zhang et al., 2019].

To construct DRS architectures, the most common practice is to design and tune the different components in a hand-crafted fashion. However, manual development is fraught with three inherent challenges. First, this requires extensive expertise in deep learning and recommender systems. Second, substantial engineering labor and time cost are required to design task-specific components for various recommendation scenarios. Third, human bias and error can result in suboptimal DRS components, further reducing recommendation performance.

Recently, powered by the advances of both theories and technologies in automated machine learning (AutoML), tremendous interests are emerging for automating the components of DRS. By involving AutoML for the deep recommender systems, different models can be automatically designed according to various data, thus improving the prediction performance and enhancing generalization. Besides, it is helpful to eliminate the negative influence for DRS from human bias and error, as well as reduce artificial and temporal costs significantly. As shown in Table 1, we summarize these researches from the perspective of search space and search strategy, which are two critical factors for AutoML. Typically, these works can be divided into the following categories according to the components in DRS:

- **Feature Selection**: This is the process of selecting a subset of the most predictive and relevant features (or generated features) for subsequent DRS models. By eliminating the redundant or irrelevant features, feature selection can help enhance the recommendation performance and accelerate DRS model training [Nadler and Coifman, 2005].

- **Feature Embedding**: Typically, the features for DRS are high-dimensional and extremely sparse. Most recommendation models first transform the raw features into one-hot vectors and then embed them as dense representations via the feature embedding layer. AutoML technique is utilized to dynamically search the optimal embedding sizes for improving prediction accuracy, saving storage space, and reducing model capacity.

- **Feature Interaction**: Effectively modeling predictive feature interactions is critical for boosting the recommendation quality of DRS because the interaction of two features would alter their individual feature effects. For example, users often download food delivery apps at mealtime, in

\*Both authors contributed equally to this research.
which interactions between app category and time-stamp is a highly predictive signal. Therefore, some AutoML-based works are devoted to exploring beneficial feature interactions with proper interaction functions.

- **System Design**: In addition to the above components of DRS models, system design also has a crucial impact on DRS performances, including hardware infrastructure, data pipeline, and information transfer, as well as implementation, deployment, optimization, and evaluation.

This survey is to provide a literature overview on the advances of AutoML for constructing DRS architectures. To be specific, we first provide an overview of AutoML techniques. Then, we discuss the state-of-the-art AutoML approaches that automate the feature selection, feature embeddings, feature interactions, and system design in DRS models. Finally, we discuss the appealing directions that can bring this research field into a new frontier.

## 2 Overview of AutoML

Given the problem description and datasets, the goal of Automated Machine Learning (AutoML) techniques is to construct machine learning solutions automatically for time-consuming and iterative real-world tasks. It has shifted the model design mechanism from hand-crafted to automatic, enabling unparalleled prospects for deep learning model construction. AutoML frameworks are typically comprised of three following components:

- **Search Space**: The search space defines a group of candidate operations and the relationships between them that enable appropriate model designs to be formed. For DRS, different components contain diverse search spaces, which are involved with human prior knowledge.

- **Search Strategy**: The search strategy specifics how to do an efficient exploration of the search space and find out the optimal architectures, which typically contains gradient-based optimization [Ruder, 2016], reinforcement learning (RL) [Kaelbling et al., 1996], evolutionary algorithms [Qin et al., 2008], Bayesian optimization [Snoek et al., 2012], random search [Bergstra and Bengio, 2012], etc.

- **Performance Estimation Strategy**: Performance estimation is the process of estimating the performance of sampled candidate architectures from the massive search space. To reduce the computational cost of training and estimating over these candidates, various strategies for performance estimation have been proposed, such as weight sharing [Pham et al., 2018] and network morphism [Elsken et al., 2017].

## 3 Feature Selection

In recommender systems, feature selection aims to select a subset of relevant features for constructing recommendation models. In practical online service providers, data is composed of a massive amount of features, including user portraits, item attributes, behavior features, contextual features as well as combinatorial features based on previous feature types. However, some of these raw features may be irrelevant or redundant in recommendations, which call for effective feature selections that can boost recommendation performance, overcome input dimensionality and overfitting, enhance model generalization and interpretability, as well as accelerate model training.

The classic feature selection methods are typically presented in three classes: 1) Filter methods, which select features based only on feature correlations regardless of the model [Hall, 1999; Yu and Liu, 2003]; 2) Wrapper methods, which evaluate subsets of features that allows detecting the possible interactions amongst variables [Maldonado and Weber, 2009]; and 3) Embedded methods, where a learning algorithm performs feature selection and classification simultaneously, such as LASSO [Fonti and Belitser, 2017] and decision trees [Ke et al., 2017]. These methods, however, usually fail in deep learning-based recommender systems with both numerical and categorical features. For instance, filter methods neglect the dependencies between feature selection and downstream deep recommendation models; Wrapper methods must explore $2^m$ candidate feature subspaces, i.e., keep or drop for $m$ feature fields; Embedded methods are sensitive to the recommendation models’ strong structural assumptions. To deal with these issues, AutoML-based methods are utilized to adaptively select effective features for recommendations. According to the feature selection stage, we categorize the research into two groups: *Selection from Raw Features* and *Selection from Generated Features*.

### 3.1 Selection from Raw Features

According to a survey from Crowdflower, scientists spend 80% of time on data and feature preparation. Therefore, introducing AutoML into raw feature selection has the potential to significantly enhance the data scientists’ productivity, and frees them up to focus on real business challenges. FSTD [Kroon and Whiteson, 2009; Fard et al., 2013] introduces
reinforcement learning (RL) into feature selection with single agent. They have a large search space of $2^m$ ($m$ is the number of feature fields), where each candidate is a possible feature subset. To limit the searching complexity, MARLFS [Liu et al., 2019b; Liu et al., 2021a] reformulates feature selection as a multi-agent reinforcement learning problem. Each feature is assigned an agent, and then all feature agents maintain to select or deselect the corresponding feature simultaneously. To reduce the computations, the following efforts attempt to accelerate the searching efficiency by learning with external knowledge [Fan et al., 2020a] and reducing the number of agents [Zhao et al., 2020c; Fan et al., 2021]. However, due to the intrinsically low sample efficiency, RL-based methods are still difficult to be integrated into real-world recommender systems with large-scale user-item interactions. To this end, AutoField [Wang et al., 2022] is proposed for practical recommendations, where the search space is relaxed to be continuous by allocating two variables to control the selection of each feature. Afterward, the selected features can be evaluated on the validation dataset by gradient descent.

### 3.2 Selection from Generated Features

In addition to selecting informative features from the raw feature set, some works learn to discover and generate beneficial combinatorial features (i.e., cross features), including categorical and statistical features. GLIDER [Tsong et al., 2020] utilizes the gradient-based neural interaction detector to detect generic non-additive and high-order combinatorial features efficiently, whose search space is $2^{c^2}$. The detected features are evaluated by a linear regression model and trained from scratch. Similarly, AutoCross [Luo et al., 2019] searches useful high-order cross features by transferring the original space to a tree-structured space, reducing the search space from $2^{c^2}$ to $(C_2^Q)^k$, where $k$ is the expected number of cross features. Then a greedy-based beam search [Medress et al., 1977] is performed to prune unpromising branches for further improving the efficiency. The feature set evaluation is achieved by field-wise logistic regression approximately and trained from scratch.

To discover useful statistical features from the raw feature set, AEF [Zhong et al., 2021] designs second-order combinatorial features search space with size $2^{c^2} - Q$, where $Q$ is the number of pre-defined construction rules. After the feature generation of groupby, aggregating, and paradigm combination, a greedy-based search with feature filtering is deployed and the selected features are evaluated from scratch.

### 4 Feature Embedding

Different from Computer Vision (CV) and Natural Language Processing (NLP), the input features used in recommender systems are extremely sparse and high-dimensional. To tackle this problem, neural network-based models leverage a feature embedding layer to map the high-dimensional features into a low-dimensional latent space. Specifically, for a feature field, we assign each feature $f_i$ with a dense embedding vector $e_i$ and save all embedding vectors in an embedding table $E \in \mathbb{R}^{V \times d}$, where $V$ and $d$ are the vocabulary size and pre-defined embedding size, respectively.

| Feature | Feature Embedding |
|---------|-------------------|
| $f_1$   | $e_1$             |
| $f_2$   | $e_2$             |
| $f_3$   | $e_3$             |
| $f_4$   | $e_4$             |
| $f_5$   | $e_5$             |
| $f_6$   | $e_6$             |
| $f_{r-1}$ | $e_{r-1}$        |
| $f_r$   | $e_r$             |

As shown in Figure 1, based on the embedding table $E$, we can obtain the embedding vectors through the embedding look-up process.

Feature embedding is the cornerstone of the DRS as the number of parameters in DRS is heavily concentrated in the embeddings and the subsequent components are constructed on the basis of feature embeddings. The feature embedding layer not only directly affects storage capacity and online inference efficiency [Guo et al., 2021], but also has a non-negligible effect on the prediction accuracy. To improve the prediction accuracy, save storage space and reduce model capacity, some AutoML-based solutions are proposed to dynamically search the embedding sizes for different features. The intuition behind is that assigning high-dimensional embeddings for high-frequency features can improve model capacity while low-dimensional embeddings for low-frequency features contribute to preventing overfitting [Zhao et al., 2020b]. According to whether or not the optimal embedding size is searched for each feature value, these solutions can be divided into categories: **Single Embedding Search** and **Group Embedding Search**, as in Figure 1.

#### 4.1 Single Embedding Search

Single embedding search-based methods [Yan et al., 2021; Liu et al., 2021b] aim to search optimal embedding dimension for each feature value, hence facing huge search space due to the large vocabulary size $V$. AMTL [Yan et al., 2021] leverages a twins-based architecture to avoid the unbalanced parameters update problem due to the different frequencies and designs an embedding search space with $d^V$, where $d$ is the embedding size. The twins-based architecture acts as a frequency-aware policy network to search the optimum dimension for each feature value, and the learning process is relaxed to a continuous space by tempered softmax [Hinton et al., 2015] and optimized by gradients. PEP [Liu et al., 2021b] proposes a pruning-based solution by enforcing column-wise sparsity on the embedding table with $L_0$ normalization, generating $2^V$ search space. To save computation cost and avoid setting pruning thresholds manually, PEP utilizes the trainable pruning parameters to prune each element automatically, which can be jointly optimized with the model parameters via gradient-based back-propagation.

However, the search space of AMTL and PEP are highly related with the embedding size $d$, which hinders the opti-
mization procedure. To reduce the search space, a solution is to divide the embedding dimension into several column-wise sub-dimensions (e.g., slicing the original dimension $d = 64$ into $\{2, 4, 8, 16, 32, 64\}$ candidate sub-dimensions). AutoEmb [Zhao et al., 2020b] and ESAPN [Liu et al., 2020c] reduce the search space from $d^V$ to $a^V$, where $a$ is the number of candidate sub-dimensions, greatly shrinking the search space compared with AMLT. AutoEmb performs a soft-selection strategy by summing over the candidate sub-dimensions with learnable weights. Instead, ESAPN performs a hard-selection strategy and a frequency-aware policy network serves as an automated RL agent to decide whether to enlarge the dimensions under the streaming setting.

Besides searching embedding dimension dynamically for each feature, learning embeddings via combination adaptively is also a trend. ANT [Liang et al., 2020] and AutoDis [Guo et al., 2021] leverage the combination over a set of anchor embeddings (named meta-embeddings in AutoDis) to represent categorical and numerical feature respectively, building a $2^{km}$ search space, where $k$ is the number of anchor embeddings. ANT uses a sparse transformation operation to hard-select relevant anchor embedding while AutoDis designs an automatic discretization network to soft-select informative meta-embeddings.

### 4.2 Group Embedding Search

AutoEmb and ESAPN shrink the search space by dividing the embedding dimension into candidate column-wise sub-dimensions. Another solution is to group the feature values of a field based on some indicators (e.g., frequencies) and assign a row-wise group embedding dimension for all the values within the group. A special case is setting the number of groups $b = 1$ and searching a global embedding dimension for all the feature values of a field, such as AutoDim [Zhao et al., 2021c]. AutoDim pre-defines several candidate sub-dimensions like AutoEmb but has a smaller search space, shrinking from $a^V$ to $a^m$. The optimization procedure is achieved by a bi-level gradient-based algorithm with Gumbel-Softmax trick [Jang et al., 2016].

To balance the search efficiency and performance, some works split the feature into multi-groups (i.e., $b > 1$) based on the feature frequencies or clustering. DNIS [Cheng et al., 2020] divides the feature values with similar frequencies into $b$ groups, reducing the search space from $2^{Vd}$ into $2^{bd}$. Then, a gradient-based differentiable search with gradient normalization is performed to search optimal group embeddings. Specifically, NIS [Joglekar et al., 2020] and RULE [Chen et al., 2021] reduce the search space significantly from both row-wise and column-wise perspectives. NIS designs single-size embedding search (with $ab$ space) and multi-size embedding search (with $b^h$), and uses the RL-based method to find the optimal embedding dimensions. Similarly, RULE divides the embedding table into multi-blocks and builds a $2^{ab}$ search space. Then the evolutionary search algorithm is proposed to search optimal item embeddings under memory constraint for the on-device recommendation. Each sub-structure is evaluated by an accurate performance estimator to balance the prediction confidence and training time.

### 5 Feature Interaction

Effectively modeling feature interactions is one of the most commonly-used approaches for DRS models to improve prediction performance. Recently, plenty of works leverage various operations to capture informative interaction signals explicitly and implicitly, such as inner product (PNN [Qu et al., 2016]), outer product (CFM [Xin et al., 2019]), convolution (FGCNN [Liu et al., 2019a]) and etc. However, these works utilize identical interaction functions to model all the feature interactions indiscriminately, which may introduce noisy interactive signals and weaken the effectiveness of modeling. To overcome these issues, some AutoML-based methods are designed to search beneficial feature interactions with optimal interaction function adaptively. These methods can be categorized into three groups depending on the search space: Feature Interaction Search, Interaction Function Search, and Interaction Block Search, as shown in Figure 2.

#### 5.1 Feature Interaction Search

To search beneficial feature interactions for enriching information, some AutoML-based works design high-order feature interactions search space and leverage search algorithms (mainly gradient-based search algorithm) to derive feature interactions automatically. AutoFIS [Liu et al., 2020b] identifies and selects important feature interactions by enumerating all the feature interactions and introducing a set of architecture parameters “gates” to indicate the importance of individual feature interactions, facing $2^{2km}$ search space even in second-order feature interactions. The architecture parameters are optimized by gradient descent with GRDA optimizer [Chao and Cheng, 2019] to get a sparse solution automatically. However, when searching high-order feature interactions, the search space of AutoFIS is huge, resulting in low search efficiency.

To solve the efficiency-accuracy dilemma, AutoGroup [Liu et al., 2020a] proposes automatic feature grouping, reducing the $p^{th}$-order search space from $2^{pkm}$ to $2^{gpm}$, where $g$ is the number of pre-defined groups. The discrete search space is then relaxed to continuous and the derivative is approximated by introducing the Gumbel-Softmax trick [Jang et al., 2016]. AutoHash [Xue et al., 2020] shares a similar idea with AutoGroup to reduce high-order search space by the hashing function.

Although AutoGroup and AutoHash improve the high-order interaction search efficiency via feature grouping and
hashing, they ignore the order-priority property [Gao et al., 2021], which reveals that the higher-order feature interactions quality can be relevant to their de-generated low-order ones, and lower-order feature interactions are likely to be more vital compared with higher-order ones. To reduce the architecture parameters and search costs, PROFIT [Gao et al., 2021] distills the \( p \)-th order search space from \( 2^{C_n^p} \) to \( 2^{m^p} \) by the composition of low-rank tensors approximately. Then to ensure the order-priority property, a progressive search algorithm based on the gradient is proposed to search high-order feature interactions order-by-order. Similarly, FIVES regards the original features as a feature graph conceptually and models the high-order feature interactions by a GNN with layer-wise adjacency matrix, so that the \( p \)-th order-search space is reduced from \( 2^{C_n^p} \) to \( 2^{m^p} \). Then, FIVES parameterizes the adjacency matrix and makes them depend on the previous layer, so that the order-priority property can be kept.

The above-mentioned works search beneficial feature interactions for all users non-personally, which overlooks the individuality and personality of the user’s behavior. To provide personalized selection of second-order feature interaction, BP-FIS [Chen et al., 2019a] designs a personalized search space with size \( 2^{ac_n^2} \), where \( a \) is the number of users. Specifically, BP-FIS proposes bayesian personalized feature interaction selection mechanism under the Bayesian Variable Selection (BVS) [Tibshirani, 1996] theory by forming a Bayesian generative model and deriving the Evidence Lower Bound (ELBO), which can be optimized by an efficient Stochastic Gradient Variational Bayes (SGVB) method.

5.2 Interaction Function Search

As suggested by PIN [Qu et al., 2018], different feature interactions are suitable for different interaction functions. Therefore, searching optimal interaction functions contributes to better capturing informative interaction signals. SIF [Yao et al., 2020] automatically devises suitable interaction functions for collaborative filtering (CF) task with two fields, which consists of micro search space referring to element-wise MLP and macro search space including 5 pre-defined operations (i.e., multiply, plus, min, max, and concat). A bi-level gradient-based search algorithm is utilized to relax the choices among operations in a continuous space. AutoFeature [Khawar et al., 2020] extends the interaction functions search to multi-field high-order scenarios by utilizing micro-networks with different architectures to model feature interactions. The whole search space expands to \( b^{C_n^2} \) for the \( p \)-th order interactions, where \( b \) is the number of candidate interaction functions, including add, Hadamard-product, concat, Generalized-product, and null. The search process is implemented by an evolutionary algorithm with the Naive Bayes tree, and each sampled architecture is trained and evaluated from scratch.

However, the interaction calculations of SIF and AutoFeature are artificially specified, which requires high dependence on domain knowledge. To overcome this limitation, AOANet [Lang et al., 2021] proposes a generalized interaction paradigm by decomposing commonly-used structures into Projection, Interction and Fusion phase. Therefore, the formula of the \( l \)-th interaction layer \( C_l \) is given as:

\[
Z_{u,v} = \{u,v\} \in \text{pairing}(B_0, B_{l-1}) \}
\]

Then architecture parameters are introduced to distinguish the importance of interaction maps, and the optimization procedure is achieved by a gradient-based method like AutoFIS.

5.3 Interaction Block Search

Searching appropriate interaction functions for different feature interactions may bring huge search space and high search overhead. Therefore, one straightforward idea is to take the original features as a whole and modularize representative operations in several blocks to formulate a generalizable search space, which is widely used in CV tasks [Liu et al., 2018]. AutoCTR [Song et al., 2020] designs a two-level hierarchical search space by abstracting the raw features and operations (i.e., MLP, FM [Rendle, 2010], and dot-product) into virtual blocks, which are further connected as a directed acyclic graph (DAG). Similar to AutoFeature, AutoCTR utilizes a multi-objective evolutionary algorithm with architectural-level learning-to-rank guidance to search the optimal architecture. The sampled architectures are evaluated from scratch and some tricks (e.g., data sub-sampling and warm-start) are used to accelerate the evaluation process.

To further improve computational efficiency, AutoPI [Meng et al., 2021] utilizes a gradient-based search strategy for exploration in a more efficient search space. AutoPI designs a hierarchical search space with blocks connecting into a DAG where the interaction cell formulates the higher-order feature interactions and the ensemble cell combines lower-order and higher-order interactions. Then, a bi-level optimization approach is applied to discover optimal architecture after the continuous relaxation.

5.4 Comprehensive Search

In addition to designing a single search space (e.g., feature embedding, feature interaction, or interaction block) in DRS, some works design a hybrid search space and perform a comprehensive search. Based on AutoFIS, AIM [Zhu et al., 2021] designs a mixed search space to select significant feature interactions, appropriate interaction functions, and optimal embedding dimensions automatically in a unified framework. It is noteworthy that the \( p \)-th order feature interaction search is achieved by combining raw features with the maintained top- \( k \) \( (p−1)^{th} \)-order feature interactions, reducing the search space from \( 2^{C_n^m} \) to \( 2^{km} \). Besides, the “gates” are extended to search embedding dimensions with \( 2^{md} \) space and interaction functions with \( 2^{kmb} \) space. AutoIAS [Wei et al., 2021] designs an integrated search space for multiple components in DRS, including feature embedding (\( a^m \) and projection (\( a^w \)), second-order feature interaction (\( a^m + a^w \)) and interaction function (\( b^{C_n^2} \)), as well as the MLP structures. An architecture generator network is trained by policy gradient and used to produce better architectures with dependency, where Knowledge Distillation (KD) [Hinton et al., 2015] is performed to enhance consistency among sub-architectures. DeepLight [Deng et al., 2021] develops an integrated search space for feature embedding (\( 2^{Vd} \)), interaction (\( 2^{ed} \)) and MLP structures by pruning redundant parameters with \( L_2 \)
penalty. Resembling DeepLight, UMEC [Shen et al., 2020] develops an integrated search space for both embedding ($2^d$) and MLP structures. Then the sparsity is achieved by $L_2$ norms, which are further reformulated as a minimax optimization problem and optimized via a gradient-based algorithm.

AMEIR [Zhao et al., 2021a] proposes an automatic behavior modeling, feature interaction exploration and MLP structure investigation solution for both sequential and non-sequential features. It is worth noting that AMEIR designs an interaction block search space (including CNN, RNN, pooling, and attention layers) for identifying sequential patterns in the user history and high-order feature interaction search space for modeling non-sequential features. The one-shot weight-sharing random search paradigm widely-used in other works is deployed to boost search efficiency.

6 System Design

Besides the aforementioned techniques on automating key components in DRS models, scientists also devise AutoML-based frameworks and training procedures from the perspective of system design.

Framework. Works in this domain mainly optimize the recommender systems from a framework perspective. DeeprRecInfra [Gupta et al., 2020] considers the inference query sizes, arrival patterns, recommendation architectures, and underlying hardware systems (GPU/CPU) to obtain the optimal infrastructure (i.e., maximizing the latency-bounded throughput), which could help reduce the latency. In terms of models’ implementation, AutoRec [Wang et al., 2020], as the first open-source platform, provides a highly-flexible pipeline for various data formation, tasks, and models in deep recommender systems.

Training. The training process is also crucial for designing reliable recommender systems. In general cases, the loss function is vital for training. GradNorm [Chen et al., 2018] and $\lambda_{\text{Opt}}$ [Chen et al., 2019b] focus on adjusting the coefficients of loss items and optimizing parameters via gradient descent. The difference is that GradNorm aims to balance different losses of multi-task while $\lambda_{\text{Opt}}$ only adjusts the regularization level. Later, Zhao et al. [Zhao et al., 2021b] proposed an adaptive loss function search framework, AutoLoss, based on a bi-level gradient-based algorithm with Gumbel-Softmax trick [Jang et al., 2016]. AutoLoss attributes the most appropriate loss function for each data example by automatically designing various loss functions, rather than adjusting coefficients only like the aforementioned works. In the scenario of knowledge transferring, system designers should figure out which parameters should be frozen to prevent overfitting on the target dataset. With Gumbel-Softmax trick [Jang et al., 2016], AutoFT [Yang et al., 2021] automatically decides whether the embedding of a field and parameters of a layer should be fine-tuned.

7 Future Directions

Although many efforts have been made to design deep recommender systems from manual to automatic, there remain a few potential opportunities for future research directions.

GNNs-based Recommendations. Graph neural networks (GNNs) have been extensively studied recently due to their impressive capability in representation learning on graph-structured data [Ma and Tang, 2021; Xu et al., 2021; Ding et al., 2021]. Recent works have been proposed to advance deep recommender systems based on GNNs techniques [Fan et al., 2019; Fan et al., 2020b; Fan et al., 2022]. Thus, exploring the combination of AutoML and GNNs provides great opportunities to further boost the performance of GNNs-based recommendation methods. A few works have been proposed to study the combination of AutoML and GNNs together in the research community. For instance, GraphNAS [Gao et al., 2020] and Auto-GNN [Zhou et al., 2019] made the very first attempt to enable the automatic design of the best graph neural architecture via reinforcement learning techniques. The main drawback of these two methods is computationally expensive because of the large search space. To tackle this challenge, SANE [Huan et al., 2021] proposes a differentiable architecture search algorithm for GNNs, where an advanced one-shot NAS paradigm is adopted to accelerate the search process. As the very first work to apply automatic NAS techniques to GNNs-based deep recommender systems, AutoGSR [Chen et al., 2022] attempts to search for the optimal GNNs architecture on GNNs-based session recommendation through a differentiable architecture search algorithm.

Multi-Modality Recommendations. In addition to historical interactions between users and items, items’ auxiliary knowledge from various modalities (e.g., visual, acoustic, and textual) has been incorporated to learn users’ preferences for providing high-quality recommendation services [Wei et al., 2019]. Hence, it is desirable to advance deep multimodal learning via automated machine learning techniques [Yin et al., 2022; Pérez-Rúa et al., 2019], so as to design an optimal algorithm for targeted tasks. For example, as the very first work on automated neural architecture search method for deep multimodal learning, the work of Multimodal Fusion Architecture Search (MFAS) aims to find accurate fusion architectures for multi-modal classification problem [Pérez-Rúa et al., 2019]. A Bilevel Multimodal Neural Architecture Search framework (BM-NAS) is proposed to learn the architectures of multimodal fusion models via a bilevel searching scheme [Yin et al., 2022].

Other Recommendation Tasks. In addition to AutoML for GNNs-based and multimodal recommendations, various important recommendation tasks are rarely explored through automated machine learning techniques, such as POI recommendations [Zhao et al., 2020a], sequential recommendations [Kang and McAuley, 2018], social recommendations [Fan et al., 2019], etc. A few works have been conducted to apply automated neural architecture search techniques for spatio-temporal prediction such as AutoST [Li et al., 2020] and AutoSTG [Pan et al., 2021], which can help design optimal neural architectures for POI recommendations. Besides, despite the success of various deep social recommendations, heavy manual work and domain knowledge is required to inherently combine user-item interactions and social relations [Fan et al., 2020b], which can be addressed by AutoML techniques.
8 Conclusion

Deep recommender systems have attracted increasing attention in both academia and industry. Besides, automated machine learning (AutoML), as one of the most promising AI techniques, has shown its great capabilities to advance deep architecture designs from manual to automatic. In this survey, we have conducted a comprehensive overview of an emerging research field: automated machine learning for deep recommender systems. Specifically, we discuss the state-of-the-art AutoML approaches that automate the feature selection, feature embeddings, feature interactions, and system design in DRS. We expect this survey can facilitate future research directions in the academic and industry community.

References

[Bergstra and Bengio, 2012] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. Journal of machine learning research, 2012.

[Chao and Cheng, 2019] Shih-Kang Chao and Guang Cheng. A generalization of regularized dual averaging and its dynamics. arXiv preprint arXiv:1909.10072, 2019.

[Chen et al., 2018] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In Proc. of ICLR, 2018.

[Chen et al., 2019a] Yifan Chen, Pengjie Ren, Yang Wang, and Maarten de Rijke. Bayesian personalized feature interaction selection for factorization machines. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019.

[Chen et al., 2019b] Yihong Chen, Bei Chen, Xiangnan He, Chen Gao, Yong Li, Jian-Guang Lou, and Yue Wang. Aopt: Learn to regularize recommender models in finer levels. In Proc. of KDD, 2019.

[Chen et al., 2021] Tong Chen, Hongzhi Yin, Yujia Zheng, Zi Huang, Yang Wang, and Meng Wang. Learning elastic embeddings for customizing on-device recommenders. arXiv preprint arXiv:2106.02223, 2021.

[Chen et al., 2022] Jingfan Chen, Guanghui Zhu, Haojun Hou, Chunfeng Yuan, and Yihua Huang. Autogra: Neural architecture search for graph-based session recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022.

[Cheng et al., 2020] Weiyu Cheng, Yanan Shen, and Linpeng Huang. Differentiable neural input search for recommender systems. arXiv preprint arXiv:2006.04466, 2020.

[Deng et al., 2021] Wei Deng, Junwei Pan, Tian Zhou, Deguang Kong, Aaron Flores, and Guang Lin. DeepLight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving. In Proc. of WSDM, 2021.

[Ding et al., 2022] Yuhui Ding, Quanming Yao, Huan Zhao, and Tong Zhang. DiffGra: Differentiable meta graph search for heterogeneous graph neural networks. In Proc. of KDD, 2021.

[Elsken et al., 2017] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. arXiv preprint arXiv:1711.04528, 2017.

[Fan et al., 2019] Wenyi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In Proc. of WWW, 2019.

[Fan et al., 2020a] Wei Fan, Kunpeng Liu, Hao Liu, Pengyang Wang, Yong Ge, and Yanjie Fu. Autofs: Automated feature selection via diversity-aware interactive reinforcement learning. In Proc. of ICDM, 2020.

[Fan et al., 2020b] Wenyi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. A graph neural network framework for social recommendations. TKDE, 2020.

[Fan et al., 2021] Wei Fan, Kunpeng Liu, Hao Liu, Ahmad Hariri, Dejing Dou, and Yanjie Fu. Autogfs: Automated group-based feature selection via interactive reinforcement learning. In Proc. of SDM, 2021.

[Fan et al., 2022] Wenyi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. Graph trend filtering networks for recommendations. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2022.

[Fard et al., 2013] Seyed Mehdi Hazrati Fard, Ali Hamzeh, and Sattar Hashemi. Using reinforcement learning to find an optimal set of features. Computers & Mathematics with Applications, 2013.

[Fonti and Belitzer, 2017] Valeria Fonti and Eduard Belitzer. Feature selection using lasso. VU Amsterdam Research Paper in Business Analytics, 2017.

[Gao et al., 2020] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In Proc. of IJCAI, 2020.

[Gao et al., 2021] Chen Gao, Yinfeng Li, Quanming Yao, Depeng Jin, and Yong Li. Progressive feature interaction search for deep sparse network. Proc. of NeurIPS, 2021.

[Gao et al., 2021a] Hui opposed Gao, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenqiang Li, and Xiaoqiang He. An embedding learning framework for numerical features in cr prediction. In Proc. of KDD, 2021.

[Gupta et al., 2020] Udit Gupta, Samuel Hsia, Vikram Saraph, Xiaodong Wang, Brandon Reagen, Guo-Yeon Wei, Hsien-Hsin S Lee, David Brooks, and Carole-Jean Wu. Deepprocs: A system for optimizing end-to-end at-scale neural recommendation inference. In Proc. of ISCA, 2020.

[Hall, 1999] Mark Andrew Hall. Correlation-based feature selection for machine learning. 1999.

[Hinton et al., 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[Huan et al., 2021] ZHAO Huan, YAO Quanming, and TU Weime. Search to aggregate neighborhood for graph neural network. In 2021 IEEE 37th International Conference on Data Engineering (ICDE), 2021.

[Jang et al., 2016] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144, 2016.

[Joglekar et al., 2020] Manas R Joglekar, Cong Li, Mei Chen, Taihui Xu, Xiaoming Wang, Jay K Adams, Pranav Khaitan, Jiashu Liu, and Qiao V Li. Neural input search for large scale recommendation models. In Proc. of KDD, 2020.

[Kaelbling et al., 1996] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. Journal of artificial intelligence research, 1996.

[Kang and McAuley, 2018] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In Proc. of ICDM, 2018.

[Ke et al., 2017] Guolin Ke, Q. Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. Proc. of NeurIPS, 2017.

[Khawar et al., 2020] Farhan Khawar, Xu Hang, Ruiming Tang, Bin Liu, Zhenqiang Li, and Xiaoqiang He. Autofeature: Searching for feature interactions and their architectures for click-through rate prediction. In Proc. of CIKM, 2020.

[Kroon and Whiteson, 2009] Mark Kroon and Shimon Whiteson. Automatic feature selection for model-based reinforcement learning in factored mdps. In Proc. of ICML, 2009.

[Lang et al., 2021] Lang Lang, Zhenlong Zhu, Xuanye Liu, Xuanxin Zhao, Jixing Li, and Minghui Shan. Architecture and operation adaptive network for online recommendation inference. In Proc. of KDD, 2021.

[Li et al., 2020] Ting Li, Junbo Zhang, Kainan Bao, Xuyuan Li, Yexin Li, and Yu Zheng. Autotop: Efficient neural architecture search for spatio-temporal prediction. 2020.

[Li et al., 2020] Paul Pu Liang, Manzil Zaheer, Yuan Wang, and Amir Ahmed. Anchor & transform: Learning sparse embeddings for large vocabularies. arXiv preprint arXiv:2003.08197, 2020.

[Liu et al., 2018] Xuexiao Liu, Karen Simonoyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018.

[Liu et al., 2019a] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. Feature generation by convolutional neural network for click-through rate prediction. In WWW, 2019.

[Liu et al., 2019b] Kunpeng Liu, Yanjie Fu, Pengfei Wang, Le Wu, Rui Bo, and Xiaolin Li. Automating feature subspace exploration via multi-agent reinforcement learning. In Proc. of KDD, 2019.

[Liu et al., 2020a] Bin Liu, Nianyuan Xue, Huifeng Guo, Ruiming Tang, Stefanos Zafeiriou, Xianghe He, and Zhenguo Li. Autogroup: Automatic feature grouping for modelling explicit high-order feature interactions in ctr prediction. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020.

[Liu et al., 2020b] Bin Liu, Chenzu Zhu, Guinlin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xianghe He, Zhenguo Li, and Yong Yu. Autofs: Automatic feature interaction selection in factorization models for click-through rate prediction. In Proc. of KDD, 2020.
