Single-shot Feature Selection for Multi-task Recommendations

Yejing Wang
City University of Hong Kong
yejing.wang@my.cityu.edu.hk

Zhaocheng Du
Huawei Noah's Ark Lab
zhaochengdu@huawei.com

Xiangyu Zhao*
City University of Hong Kong
xianzhao@cityu.edu.hk

Bo Chen
Huawei Noah's Ark Lab
chenbo116@huawei.com

Huifeng Guo*
Huawei Noah's Ark Lab
huifeng.guo@huawei.com

Ruiming Tang
Huawei Noah's Ark Lab
tangruiming@huawei.com

Zhenhua Dong
Huawei Noah's Ark Lab
dongzhenhua@huawei.com

ABSTRACT

Multi-task Recommender Systems (MTRSs) has become increasingly prevalent in a variety of real-world applications due to their exceptional training efficiency and recommendation quality. However, conventional MTRSs often input all relevant feature fields without distinguishing their contributions to different tasks, which can lead to confusion and a decline in performance. Existing feature selection methods may neglect task relations or require significant computation during model training in multi-task setting. To this end, this paper proposes a novel **S**ingle-shot **F**eature **S**election framework for MTRSs, referred to as MultiSFS, which is capable of selecting feature fields for each task while considering task relations in a single-shot manner. Specifically, MultiSFS first efficiently obtains task-specific feature importance through a single forward-backward pass. Then, a data-task bipartite graph is constructed to learn field-level task relations. Subsequently, MultiSFS merges the feature importance according to task relations and selects feature fields for different tasks. To demonstrate the effectiveness and properties of MultiSFS, we integrate it with representative MTRS models and evaluate on three real-world datasets. The implementation code is available online to ease reproducibility[1,2].

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Feature Selection, Multi-task Learning, Recommender Systems

[1]https://gitee.com/mindspore/models/tree/master/research/recommend/MultiSFS
[2]https://github.com/Applied-Machine-Learning-Lab/MultiSFS

*Xiangyu Zhao and Huifeng Guo are corresponding authors.

ACM Reference Format:
Yejing Wang, Zhaocheng Du, Xiangyu Zhao*, Bo Chen, Huifeng Guo*, Ruiming Tang, and Zhenhua Dong. 2023. Single-shot Feature Selection for Multi-task Recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539618.3591767

1 INTRODUCTION

Deep Recommender Systems (DRS) play a critical role in today's information explosion, as they are able to capture user preferences and deliver the most appropriate items [13, 16, 36, 44]. In order to attract users and increase profits, companies are increasingly seeking to optimize multiple tasks simultaneously. For instance, online video-sharing platforms [11, 52] not only consider whether users will click on recommended videos but also whether they genuinely enjoy the video (e.g., by clicking the 'Like' or 'Favorite' buttons, or sharing the video with friends). As a result, the simultaneous prediction of multiple tasks is becoming an increasingly important research area [29, 31, 32, 40, 50]. Deploying multi-task recommender systems (MTRSs) can capture the shared information for tasks, improving training efficiency and model accuracy.

The information-sharing strategies of existing MTRSs can be broadly categorized into two groups, i.e., architecture-sharing and output-sharing. Architecture-sharing models [7, 30, 31, 33, 37, 40] typically share a sort of bottom parameters to learn common features across all tasks. These common features are then used to generate task-specific predictions through separate tower networks. A famous architecture-sharing strategy is the shared-bottom framework [7], where the entire bottom network is shared. Variations include sub-networks (experts) that share knowledge between tasks with predefined connections [33, 37] or a searched route [30]. Additionally, gating mechanisms to model task differences and balance shared parameters are also utilized to mitigate negative transfer issue [31, 40]. The other sharing strategy, output-sharing, is designed for cascading tasks, where each task depends on the previous one, such as "impression → click → conversion" [32]. These models usually use task-specific bottom networks and only share the output of task towers with the next task [32, 43, 46, 48, 50]. For example, Ma et al. [32] use the element-wise product of neighboring predictions and Xi et al. [50] design adaptive information transfer and behavior expectation calibration for modeling sequential dependence. However, both strategies neglect the different contributions of input features to tasks, i.e., all tasks are predicted with the same input feature fields[3]. In addition, some shared features might be redundant for specific tasks and thus degrade model performance [8, 9]. As an example, the video attribute "music type" may be important for determining user preference in tasks "Like" or "Hate", but may not be available before the user click and is therefore redundant for the task "Click". As a result, the selection of suitable feature subsets for different tasks in MTRSs is highly desirable in practice.

The problem of feature selection for multi-task recommendations is challenging due to the large search space. With T tasks and N feature fields, there are 2^{NT} possible selection options, making it computationally infeasible to train a model for each selection option and test its performance. One solution is to apply feature selection methods for each task individually [24, 42]. However, this approach ignores the correlation between tasks, resulting in sub-optimal feature selection results. Recent research has focused on developing feature selection methods for MTRSs [8, 9]. For instance, MSSM [9] proposes a field-level sparse connection module that enables different task networks to learn from task-specific feature fields by distributing sparse masks. Likewise, CFS [8] is a feature selection framework for MTRSs based on causal inference that aims to address negative transfer. However, MSSM and CFS still retain embedding parameters for deselected feature fields and introduce additional parameters to select features during training, making the selection less efficient. Besides, they still independently select features for each task, neglecting the connections among tasks. Thus, it is desirable to develop more efficient feature selection methods for MTRSs with considering task relations.

In this paper, we propose an efficient feature selection method for MTRSs, called **S**ingle-shot **F**eature **S**election for **Multi**-task recommendations (MultiSFS). To address the challenging problem of explicitly task relations modeling, we construct a bipartite graph [6] between tasks and data points. To be specific, MultiSFS first obtains task-specific feature importance with a single forward-backward pass by single-shot salience scoring method [18, 19]. Then, it learns task similarity from the constructed data-task graph and guide the feature selection by merging the feature importance between tasks according to their similarity. It is noteworthy that MultiSFS learns the feature field-level task similarity to merge feature importance of different feature fields with different coefficients. The major contributions of this work could be summarized as:

- We formulate the feature selection problem for multi-task recommendations with the objective of selecting predictive features prior to model training;
- We propose a novel feature selection framework for MTRSs, MultiSFS, which is capable of selecting predictive features in a task-specific manner with a single forward-backward pass. This makes MultiSFS the first framework to conduct feature selection in a single-shot manner;
- To explicitly learn task relations, we present a novel approach to modeling field-wise task relations, which has not been addressed in previous works;
- Extensive experiment results on three real-world datasets validate the effectiveness of MultiSFS. We compare its performance and efficiency with the state-of-the-art MTRSs and feature selection methods with encouraging results.

[3]A feature field is a set of feature values of the same category. For example, the "gender" feature field may contain two feature values, i.e., female and male.

2 PROBLEM FORMULATION

We formulate the feature selection problem for MTRSs in this section. For a typical multi-task recommendation dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ where $\mathcal{X}$ is the feature set and $\mathcal{Y}$ is the label set, we denote a record of user behaviors as $(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}$:

$$(\boldsymbol{x}, \boldsymbol{y}) = ([\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \ldots, \boldsymbol{x}_N], [y_1, \ldots, y_t, \ldots, y_T]) \tag{1}$$

$[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \ldots, \boldsymbol{x}_N]$ are raw feature values for N feature fields in the format of one-hot vectors, $[y_1, \ldots, y_t, \ldots, y_T]$ are binary[4] labels indicating the ground-truth of T tasks. Existing MTRSs usually input all features to predict all tasks without distinguishing their contributions [7, 30, 31, 40, 50], which could be formulated as:

$$\boldsymbol{e}_n = \boldsymbol{V}_n \boldsymbol{x}_n \tag{2}$$

$$\boldsymbol{E} = \text{Embedding}(\boldsymbol{x}) \overset{\text{def}}{=} [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N] \tag{3}$$

$$\hat{\boldsymbol{y}} = f(\boldsymbol{E}; \Theta) = [f_1(\boldsymbol{E}; \Theta_1), \ldots, f_T(\boldsymbol{E}; \Theta_T)] = [\hat{y}_1, \ldots, \hat{y}_T] \tag{4}$$

Equation (2) converts one-hot feature vectors $\boldsymbol{x}_n$ to dense low-dimensional vectors $\boldsymbol{e}_n$ by looking up the corresponding embeddings from the embedding table $\boldsymbol{V}_n$. Equation (3) defines the operation Embedding$(\boldsymbol{x})$, which concatenates embedding vectors of input features $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$. In Equation (4), f is the applied MTRS with learnable parameters $\boldsymbol{\Theta}$, and f_t is a part of the whole network f for predicting $\hat{y}_t$[5].

However, previous works [8, 9] suggest that sharing all features as Equation (4) leads to the negative transfer problem in input layers while selecting individual feature sets for tasks helps to relieve this problem. Motivated by this idea and the impressing result of saliency scores [2, 19] for pruning, we construct the feature selection as a pruning problem, which selects features by finding a binary mask matrix $\boldsymbol{M}$. We first define the following notations:

$$\tilde{\boldsymbol{E}}_t = [m_{t1}, \ldots, m_{tn}] \odot \boldsymbol{E} = [m_{t1}\boldsymbol{e}_1, \ldots, m_{tn}\boldsymbol{e}_n] \tag{5}$$

$$\hat{\boldsymbol{y}} = [f_1(\tilde{\boldsymbol{E}}_1; \Theta_1), \ldots, f_T(\tilde{\boldsymbol{E}}_T; \Theta_T)] \overset{\text{def}}{=} f(\boldsymbol{M} \odot \boldsymbol{E}; \Theta) \tag{6}$$

$$\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \mathcal{L}(f(\boldsymbol{M} \odot \boldsymbol{E}; \Theta), \boldsymbol{y}) = \sum_{t=1}^{T} \mathcal{L}_t(\hat{y}_t, y_t) \tag{7}$$

where $\{\mathcal{L}_t\}_{t=1}^{T}$ (loss functions) are predefined for T tasks, respectively. $\boldsymbol{M} \in \mathbb{R}^{T \times N}$ is the binary mask matrix, where each element m_{tn} denotes whether the n^{th} feature field is selected ($m_{tn} = 1$) or deselected ($m_{tn} = 0$) for task t. In Equation (5), we replace the embedding of deselected feature fields with all-zero vectors. For example, the embedding matrix for task t with feature field #2 and #4 dropped is denoted as $\tilde{\boldsymbol{E}}_t = [\boldsymbol{e}_1, \boldsymbol{0}, \boldsymbol{e}_3, \boldsymbol{0}, \boldsymbol{e}_5, \ldots, \boldsymbol{e}_N]$.

[4]We focus on CTR prediction (binary classification) task in this paper for simplicity, but it is straightforward to extend our model to other recommendation tasks.
[5]Note that $\{f_t\}$ are not independent for most MTRSs. We split the whole network into $\{f_t\}$ for clarity.

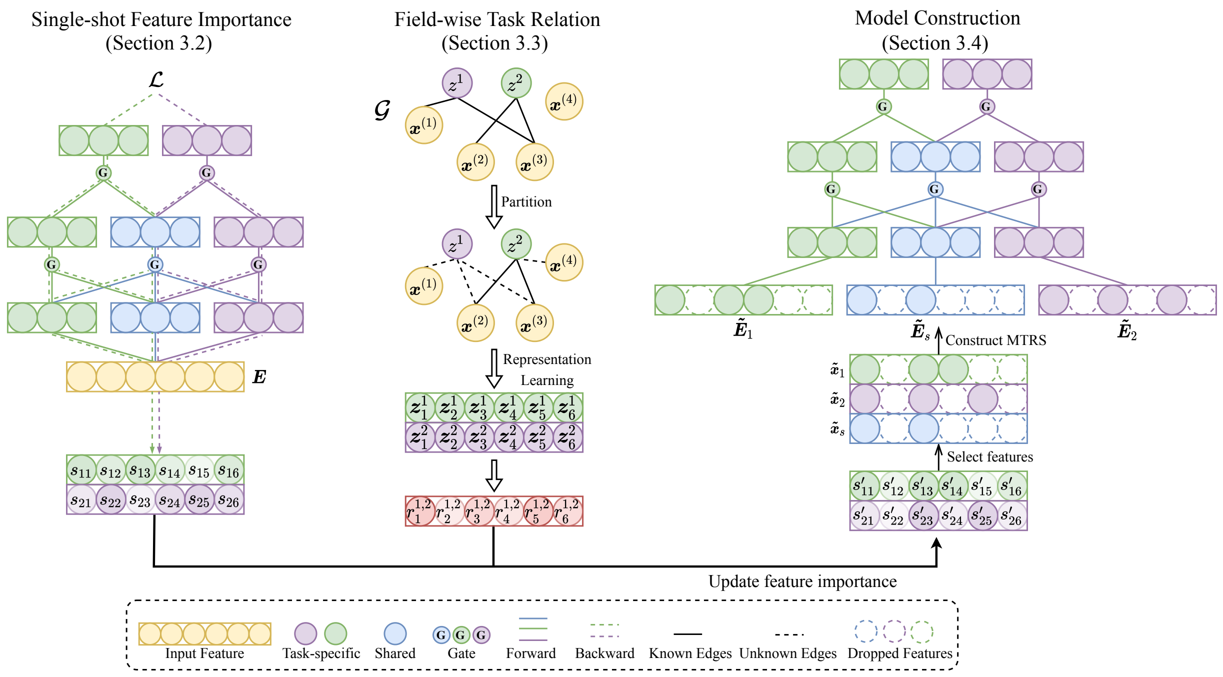

Figure 1: Framework overview, an example of PLE [40] with MultiSFS.

With the notations defined above, our feature selection problem for MTRS is formally defined as:

$$\boldsymbol{M}^* = \text{argmin}_{\boldsymbol{M}} \mathcal{L}(f(\boldsymbol{M} \odot \boldsymbol{E}; \boldsymbol{\Theta}), \boldsymbol{y}) \tag{8}$$
$$\text{s.t.} \sum_{n=1}^{N} m_{tn} = k_t, \quad m_{tn} \in \{0, 1\}$$

where hyper-parameter k_t is the number of selected feature fields for task t. Notably, $\boldsymbol{E}$ and $\boldsymbol{\Theta}$ are randomly initialized without training since we aim to decide the feature selection result before constructing MTRSs rather than selecting feature fields during the model training, so as to completely eliminate the negative effects from suboptimal feature fields in training MTRSs, differing from previous feature selection works for multi-task learning [8, 9, 28, 53].

3 FRAMEWORK

This section will first give an overview of the proposed MultiSFS. Then, we detail each part of MultiSFS, including single-shot feature importance computation and field-wise task relation learning. Finally, we introduce how to construct multi-task deep recommendation models with selected feature fields.

3.1 Framework Overview

The framework overview is shown in Figure 1. We visualize the feature selection for PLE [40] with 2 tasks and 6 feature fields as an example. Briefly, we select features for MTRS in three steps, i.e., single-shot feature importance calculation (left), field-wise task relation learning (middle), and model construction (right).

To be specific, we first calculate task-specific feature importance with a single forward-backward pass. In this step, the model takes the shared feature embeddings $\boldsymbol{E}$ as the input and generates predictions for a mini-batch of data. Then, the feature importance $\{s_{tn}\}$ is calculated based on the partial gradient of mask matrix $\boldsymbol{M}$ on $\mathcal{L}$.

Next, field-wise task relations are learned through node representation learning in the data-task bipartite graph. In this step, we construct a data-task graph where the existence of edges is determined by task labels. We randomly partition the edges into known and unknown groups and thus train node representations with GNN layers. The task relations $\{r_n^{t_1,t_2}\}$ are computed at the feature level based on task node representations $\{\boldsymbol{z}_n^t\}$.

Finally, we update the feature importance and construct MTRSs with selected features. Taking PLE in Figure 1 as an example, we select the feature fields with the highest importance score (presented with high opacity) as task-specific features (field#1,#3,#4 for task 1 and field#1,#3,#5 for task 2) and their intersection as shared features (field#1,#3). Corresponding embedding matrices $\tilde{\boldsymbol{E}}_t$ are fed into subsequent models.

3.2 Single-shot Feature Importance

In this section, we detail the single-shot feature importance computation based on their influences on loss values.

Optimizing Problem (8) can be challenging due to the binary nature of $\boldsymbol{M}$, containing only discrete values of 0, 1. To address this, we draw inspiration from SNIP [19], which relaxes the binary mask and calculates the partial gradient of the mask on the final loss to identify the importance of each model parameter. Likewise, we compute task-specific feature importance by measuring the impact of each m_{tn} on the loss value $\mathcal{L}$. Instead of solving Problem (8), we select the k_t most important feature fields for task t. The feature importance for task t is determined by the difference in loss value when dropping the n^{th} feature field:

$$\Delta\mathcal{L}_{tn} = \mathcal{L}(f(\mathbf{1} \odot \boldsymbol{E}; \Theta), \boldsymbol{y}) - \mathcal{L}(f((\mathbf{1} - \boldsymbol{\psi}_{tn}) \odot \boldsymbol{E}; \Theta), \boldsymbol{y}) \tag{9}$$

where $\mathbf{1}$ is all-one matrix in shape of $N \times T$ and $\boldsymbol{\psi}_{tn}$ is a binary matrix with 1 on the n^{th} column of the t^{th} row and zeros elsewhere.

However, it is tedious and expensive to compute every $\Delta\mathcal{L}_{tn}$ since this process requires $T \times N$ forward passes. Equation (9) aims to measure the influence of m_{tn} on the loss value by computing the difference in a discrete way. It is straightforward for us to approximate $\Delta\mathcal{L}_{tn}$ by relaxing the binary constraint on m_{tn} to a continuous range $[0, 1]$ and compute derivatives at $\boldsymbol{M} = \mathbf{1}$ as task-specific feature importance $\{s_{nt}\}$ [18, 19, 35]:

$$\Delta\mathcal{L}_{tn} \approx s_{tn} = \frac{\partial \mathcal{L}(f(\boldsymbol{M} \odot \boldsymbol{E}; \Theta)}{\partial m_{tn}} \Big|_{\boldsymbol{M}=\mathbf{1}}$$
$$= \lim_{\delta \to 0} \frac{\mathcal{L}(f(\boldsymbol{M} \odot \boldsymbol{E}; \Theta), \boldsymbol{y}) - \mathcal{L}(f((\boldsymbol{M} - \delta\boldsymbol{\psi}_{tn}) \odot \boldsymbol{E}; \Theta), \boldsymbol{y})}{\delta} \Big|_{\boldsymbol{M}=\mathbf{1}} \tag{10}$$

where δ is infinitesimal. This enables us to simultaneously evaluate the contribution of $\{m_{tn}\}_{t \leq T, n \leq N}$ to $\mathcal{L}$ in a single forward-backward pass by computing $\frac{\partial \mathcal{L}}{\partial \boldsymbol{M}}$ on a mini-batch of dataset $\mathcal{D}_{\mathcal{B}}$. Notably, a large s_{nt} indicates a significant impact on loss value $\mathcal{L}$, which implies feature field n is important to task t.

Discussion. Compared with previous works [19, 35], our method is different from the following perspectives. (***i***) We do not require the optimized MTRS parameters ($\boldsymbol{E}, \Theta$) and minimal loss values, i.e., we compute feature importance without training, which differs with SSEDS [35]. The reason is that SSEDS [35] requires fully exploring the potential of each embedding dimension and accurately distinguishing redundant ones. While we tend to discover the most predictive feature fields. In this case, scoring feature fields before training contributes to the fair comparison since the training process unequally explores the value of feature fields. (***ii***) Unlike [19, 35], we directly use derivatives as the feature importance without normalization. This is because we aim to obtain task-specific feature importance independently and consider field-wise task relations in the next step. Normalizing $\{s_{tn}\}$ for all t and n would destroy the independence from both task-wise and field-wise perspectives. (***iii***) We design the indicator parameter $\boldsymbol{M}$ from a coarser perspective to conduct feature selection. Specifically, the pruning method [19] assigns a binary mask to each model weight, which is equivalent to each element in every embedding table $\boldsymbol{V}_n$ in this paper. SSEDS [35] assigns the mask to each column of every $\boldsymbol{V}_n$. For our MultiSFS, as in Problem (8), we use $\{m_{tn}\}$ to indicate the selection result of feature field n for task t, i.e., one mask for the whole $\boldsymbol{V}_n$.

3.3 Field-wise Task Relation

The current attempts to capture task relations [4, 55] have mainly concentrated on utilizing the correlations among task labels. However, this method poses difficulties in the context of MTRSs, given the scarcity of positive labels, which results in low correlations between tasks. Moreover, it is important to note that different fields may exhibit varying relationships between tasks. A shared correlation for all fields may fail to capture this informative difference. Therefore, we propose a novel approach to capturing field-wise task relations, which enables us to better discern the contribution of each feature field and overcome these limitations.

This section details how to obtain field-wise task relations and accordingly update feature importance. We suggest that task relations should be invariant no matter which MTRS is applied.

To be specific, we first construct a data-task graph [6] $\mathcal{G} = (\{\mathcal{X}, \mathcal{Z}\}, \mathcal{Y})$, where $\mathcal{X} = \{\boldsymbol{x}^{(i)}\}_{i \leq |\mathcal{D}|}, \mathcal{Y} = \{\boldsymbol{y}^{(i)}\}_{i \leq |\mathcal{D}|}$ are data point set and task label set of dataset $\mathcal{D}$, respectively. Similar to Equation (1), $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ denote features and task labels for the i^{th} sample in $\mathcal{D}$:

$$\boldsymbol{x}^{(i)} = [x_1^{(i)}, \ldots, x_n^{(i)}, \ldots, x_N^{(i)}]$$
$$\boldsymbol{y}^{(i)} = [y_1^{(i)}, \ldots, y_t^{(i)}, \ldots, y_T^{(i)}]$$

There are two types of nodes in $\mathcal{G}$: data nodes $\mathcal{X} = \{\boldsymbol{x}^{(i)}\}_{i \leq |\mathcal{D}|}$ and task nodes $\mathcal{Z} = \{z^t\}_{t \leq T}$. The edges in $\mathcal{G}$ are determined by task labels $\mathcal{Y}$, i.e., there is an edge between $\boldsymbol{x}^{(i)}$ and z^t if and only if $y_t^{(i)} = 1$. For convenience, we use $\mathcal{Y}$ to denote the edge set of $\mathcal{G}$.

After constructing the data-task graph, we aim to learn field-wise task relations from its structure. We first initialize node representations. For data node $\boldsymbol{x}^{(i)}$, the node representation is defined as its feature embeddings $\boldsymbol{E}^{(i)}$:

$$\boldsymbol{E}^{(i)} = \text{Embedding}(\boldsymbol{x}^{(i)}) = [\boldsymbol{e}_1^{(i)}, \ldots, \boldsymbol{e}_N^{(i)}] \in \mathbb{R}^{N \times D} \tag{11}$$

where Embedding$(\cdot)$ is the same as Equation (2) but with different embedding parameters $(\boldsymbol{V}')$[6]. Here we introduce D as the unified embedding dimension for all feature fields.

The node representation of task node z^t is randomly initialized:

$$[z_1^t, \ldots, z_n^t, \ldots, z_N^t] \in \mathbb{R}^{N \times D} \tag{12}$$

where z_n^t has the same shape as $\boldsymbol{e}_n^{(i)}$, i.e., $\mathbb{R}^D$. In order to learn field-wise task relations, Equation (12) separately initializes the task node embedding from the field level rather than directly generating a long vector with the length ND. This initialization method contributes to constructing field-wise connections between data nodes and task nodes, which shares a similar idea to intent-aware recommendations [45, 49].

To learn embedding parameters $\boldsymbol{V}'$ and task node representations $\{z_n^t\}_{t \leq T, n \leq N}$, we construct the following training task:

$$\left(\boldsymbol{x}^{(i)}, \{y_j^{(i)}\}_{j \in Aux(i)}\right) \rightarrow \{y_j^{(i)}\}_{j \in Test(i)} \tag{13}$$

where $Aux(i)$ and $Test(i)$ are randomly partitioned, indicating indices of given auxiliary task labels and unknown test task labels, respectively. Note that $Aux(i) \cap Test(i) = \emptyset, Aux(i) \cup Test(i) = [1, \ldots, T]$ and partitions are not the same for all samples, i.e., $Aux(1)$ and $Aux(2)$ can be different. The detailed partition could refer to relational learning in [6].

Next, we learn field-wise task relations from $\mathcal{G}$ via a heterogeneous GNN. For the l^{th} layer, we denote hidden states of data nodes as $\{\boldsymbol{E}_l^{(i)}\}_{i \leq |\mathcal{D}|}$ and task nodes as $\{Z_l^t\}_{t \leq T}$. The initial state could be formulated as:

$$\boldsymbol{E}_0^{(i)} = \boldsymbol{E}^{(i)} \in \mathbb{R}^{N \times D} \tag{14}$$

$$Z_0^t = Z^t \in \mathbb{R}^{N \times D} \tag{15}$$

[6]The reason is that task relation learning and feature importance computation are independent parts. We denote the embedding table as $\boldsymbol{V}' = \{\boldsymbol{V}_n'\}_{n \leq N}$ for distinguishing.

Algorithm 1 Field-wise Task-relation Learning

```
Input: Training data $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$
Output: Field-wise task relations $\{r_n^{i,j}\}$
1: Construct data-task graph $\mathcal{G}$;
2: Initialize node representations with parameters $V^*, \{z_n^t\}$;
3: Randomly partition $Aux(i)$ and $Test(i)$;
4: Train the GNN by minimizing $\mathcal{L}_\mathcal{G}$ in Equation (22);
5: Compute field-wise task relations $\{r_n^{i,j}\}$ as Equation (23);
```

Algorithm 2 The proposed MultiSFS

```
Input: Training data $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$, MTRS $f$, feature numbers $\{k_t\}$, loss functions $\{\mathcal{L}_t\}$, field-wise task relations $\{r_n^{i,j}\}$
Output: $f$ with selected feature
1: Sample a mini-batch $\mathcal{D}_b$ from $\mathcal{D}$;
2: Compute $\{s_{tn}\}$ as Equation (10) based on $\mathcal{D}_b$;
3: Update feature importance $\{s_{tn}\}$ with $\{r_n^{i,j}\}$ as Equation (24) ;
4: Decide $\{\tilde{x}_t\}$ and $\tilde{x}_s$;
5: Construct $f$ with selected feature fields.
```

where $\boldsymbol{E}^{(i)}$ and $\boldsymbol{Z}^t$ are node representations defined in Equation (11) and Equation (12). For subsequent layers, we propose an extended version of GraphSAGE [6, 15]:

$$\boldsymbol{E}_l^{(i)} = \boldsymbol{U}_l[\text{Mean}(\{\text{ReLU}(\boldsymbol{W}_l^t \boldsymbol{Z}_{l-1}^t), z^t \in \mathcal{N}(\boldsymbol{x}^{(i)})\}), \boldsymbol{E}_{l-1}^{(i)}] \quad (16)$$

$$\boldsymbol{Z}_l^t = \boldsymbol{U}_l[\text{Mean}(\{\text{ReLU}(\boldsymbol{W}_l^d \boldsymbol{E}_{l-1}^{(i)}), \boldsymbol{x}^{(i)} \in \mathcal{N}(z^t)\}), \boldsymbol{Z}_{l-1}^t] \quad (17)$$

where $\boldsymbol{U}_l \in \mathbb{R}^{N\times D\times 2D}, \boldsymbol{W}_l^t, \boldsymbol{W}_l^d \in \mathbb{R}^{N\times D\times D}$ are learnable parameters. $\mathcal{N}(\cdot)$ denotes the direct neighbor set of the specific node, which is determined by given task labels $\{y_j^{(i)}\}_{j\in Aux(i)}$.

Notable, there are two extensions in our GraphSAGE layers. First, we design bi-directional message-passing functions. Specifically, we use $\boldsymbol{W}_l^t$ and $\boldsymbol{W}_l^d$ to pass the message from task nodes to data nodes and vice versa, respectively. This helps to learn different types of messages [6]. Second, we conduct the message passing at the field-level (row-wise) in order to keep the independence across feature fields. Specifically, each row of the updated hidden state only depends on the same row of itself and its neighbors' states at the previous layer, together with corresponding groups of parameters. For example, the n^{th} row of $\boldsymbol{E}_l^{(i)}$, denoted as $\boldsymbol{e}_{l,n}^{(i)}$, is calculated as:

$$\boldsymbol{e}_{l,n}^{(i)} = \boldsymbol{U}_{l,n}[\text{Mean}(\{\text{ReLU}(\boldsymbol{W}_{l,n}^t z_{l-1,n}^t), z^t \in \mathcal{N}(\boldsymbol{x}^{(i)})\}), \boldsymbol{e}_{l-1,n}^{(i)}] \quad (18)$$

where $\boldsymbol{U}_{l,n} \in \mathbb{R}^{D\times 2D}, \boldsymbol{W}_{l,n}^t \in \mathbb{R}^{D\times D}$ are the n^{th} slice of $\boldsymbol{U}_l, \boldsymbol{W}_l^t$ at the first dimension. The calculation is similar for $z_{l,n}^t$.

After L layers of GNN, we obtain updated node representations:

$$\boldsymbol{E}_L^{(i)} = [\boldsymbol{e}_{L,1}^{(i)}, \dots, \boldsymbol{e}_{L,N}^{(i)}] \quad (19)$$

$$\boldsymbol{Z}_L^t = [z_{L,1}^t, \dots, z_{L,N}^t] \quad (20)$$

To predict $\{y_j^{(i)}\}_{j\in Test(i)}$, we sum up field-level scores with the activation function Sigmoid$(\cdot)$:

$$\hat{y}_j^{(i)} = \sum_{n=1}^{N} \text{Sigmoid}(\boldsymbol{e}_{L,n}^{(i)} \cdot z_{L,n}^j) \quad (21)$$

Task node representations $\{z_n^t\}_{t\le T, n\le N}$ are finally learned by minimizing binary-cross-entropy (BCE) loss $\mathcal{L}_\mathcal{G}$:

$$\mathcal{L}_\mathcal{G} = \sum_{i\le|\mathcal{D}|} \sum_{j\in Test(i)} y_j^{(i)} \cdot \log \hat{y}_j^{(i)} + \left(1 - y_j^{(i)}\right) \cdot \log\left(1 - \hat{y}_j^{(i)}\right) \quad (22)$$

After the GNN training, we compute field-wise task relations based on task node representations $\{z_n^t\}$. Take the task pair t_1 and t_2 for example, the field-wise task relation is computed as:

$$r_n^{t_1,t_2} = r_n^{t_2,t_1} = \begin{cases} \dfrac{z_n^{t_1} \cdot z_n^{t_2}}{||z_n^{t_1}||_2 ||z_n^{t_2}||_2}, & t_1 \neq t_2 \\ 1, & t_1 = t_2 \end{cases} \quad (23)$$

Finally, the feature importance $\{s_{tn}\}_{t\le T, n\le N}$ in Equation (10) is updated based on the field-wise task relations $\{r_n^{i,j}\}_{i,j\le N}$ as presented in Equations (23). The intuition behind this update is that tasks that are highly related from the perspective of a particular feature field tend to make similar decisions regarding the selection or deselection of that feature. Thus, the feature importance is modified through a weighted sum, as expressed in Equation (24):

$$s'_{tn} = \sum_{i=1}^{T} r_n^{t,i} s_{tn} \quad (24)$$

3.4 MTRS Construction

In this section, we introduce how to select feature fields for tasks according to feature importance $\{s'_{tn}\}_{t\le T, n\le N}$. As visualized in Figure 1, we need to determine two sets of selected features, task-specific feature sets $\{\tilde{x}_t\}_{t\le T}$ and the shared feature set $\tilde{x}_s$.

As described in Section 2, we select task-specific feature fields by deciding the binary matrix $\boldsymbol{M} \in \mathbb{R}^{T\times N}$, i.e., feature field n is selected for task t if $m_{tn} = 1$. For every task t, to fulfill the constrain $\sum_{n=1}^{N} m_{tn} = k_t$ in Problem (8), we set $m_{tn} = 1$ if the corresponding score s'_{tn} is larger than k_t-th largest in $[s_{t1}, \dots, s_{tN}]$, or $m_{tn} = 0$.

To decide the shared feature set, we could take the union or intersection of selected feature fields, i.e., $\tilde{x}_s = \bigcup_{t\le T} \tilde{x}_t$ or $\tilde{x}_s = \bigcap_{t\le T} \tilde{x}_t$. In practice, we choose one of the solutions according to the characteristics of the MTRS used. For example, we take $\tilde{x}_s = \bigcup_{t\le T} \tilde{x}_t$ for MMoE [31] since there are no task-specific experts and we only input the shared feature set to the model. For PLE [40], the example in Figure 1, $\tilde{x}_s = \bigcap_{t\le T} \tilde{x}_t$ is used since it designs task-specific experts and shared experts for each task. Taking the interaction helps the shared experts of PLE to focus on common features of tasks. For MTRSs like ESMM [32] or AITM [50], we could just skip this step since there are no shared bottom architectures for tasks.

The detailed steps of our MultiSFS are summarized in Algorithm 2. The feature importance $\{s_{tn}\}$ is first computed with a mini-batch of data $\mathcal{D}_b$ in a single forward-backward pass (line 1 - 2). Then, with field-wise task relations $\{r_n^{i,j}\}$ learned from Algorithm 1, the feature importance is updated (line 3). We finally select task-specific feature sets and the shared features to construct MTRS f (line 4 - 5). The whole process is single-shot.

Table 1: Overall performance improvement by applying MultiSFS to MTRS backbones.

Dataset	KuaiRand-Pure				QB-Video				AliCCP			
Metric	AUC1↑	AUC2↑	Logloss1↓	Logloss2↓	AUC1↑	AUC2↑	Logloss1↓	Logloss2↓	AUC1↑	AUC2↑	Logloss1↓	Logloss2↓
Single Task	0.6942	0.8571	0.6401	0.0765	0.9794	0.9013	0.1520	0.0295	0.6142	0.6130	0.1601	0.2141
MMoE	0.6914	0.8613	0.6416	0.0766	0.9792	0.9013	0.1527	0.0290	0.6155	0.6334	0.1597	0.2122
MMoE + MultiSFS	**0.6929***	**0.8628***	**0.6395***	**0.0765**	**0.9793**	**0.9044***	**0.1523**	**0.0286**	**0.6182***	**0.6389***	**0.1595**	**0.2099***
Gain (%)	0.21	0.17	0.32	0.08	0.0	0.34	0.28	1.45	0.44	0.88	0.13	1.09
PLE	0.6916	0.8600	0.6412	0.0767	0.9792	0.8923	0.1519	0.0294	0.6152	0.6377	0.1595	0.2112
PLE + MultiSFS	**0.6920**	**0.8645***	**0.6405**	**0.0755***	**0.9794**	**0.9072***	0.1524	**0.0283***	**0.6164***	**0.6420***	**0.1592**	**0.2091***
Gain (%)	0.06	0.53	0.10	1.50	0.02	1.66	-0.33	3.67	0.19	0.67	0.16	1.01
ESMM	0.6919	0.8598	0.6411	0.0764	0.9793	0.9048	0.1522	0.0291	0.6142	0.6292	0.1600	0.2141
ESMM + MultiSFS	**0.6941***	**0.8622***	**0.6406**	**0.0761**	**0.9794**	**0.9068***	**0.1516**	**0.0288**	**0.6181***	**0.6439***	**0.1590***	0.2142
Gain (%)	0.31	0.28	0.08	0.44	0.02	0.22	0.38	0.89	0.63	2.33	0.62	0.03
AITM	0.6916	0.8589	0.6413	0.0770	0.9793	0.9008	0.1521	0.0289	0.6143	0.6325	0.1599	0.2151
AITM + MultiSFS	**0.6946***	**0.8659***	**0.6407**	**0.0761**	**0.9797**	**0.9065***	**0.1511**	**0.0288**	**0.6154***	**0.6338***	**0.1596**	0.2163
Gain (%)	0.43	0.81	0.10	1.12	0.04	0.63	0.60	0.45	0.17	0.20	0.23	-0.57

"*" indicates the statistically significant improvements (i.e., paired t-test with $p < 0.05$) over the original model.

Table 2: Dataset statistics

	# User	# Item	# Interactions	#Feature
KuaiRand-Pure	25,970	7,280	955,384	41
QB-Video	34,240	130,637	2,442,299	12
AliCCP	138,326	310,057	52,439,671	17

4 EXPERIMENT

This section details our experiment settings and presents extensive experimental results to figure out the following research questions:

- **RQ1:** How does the proposed framework, MultiSFS, perform compared with MTRSs models and feature selection baselines?
- **RQ2:** Does MultiSFS still works with various multi-task settings?
- **RQ3:** How is the efficiency of MultiSFS?
- **RQ4:** What is the impact of core components in MultiSFS?
- **RQ5:** How do hyper-parameters influence MultiSFS?

4.1 Experiment Settings

4.1.1 Dataset. We evaluate the performance of MultiSFS on three public datasets. **KuaiRand-Pure [11]** is collected from one popular Chinese video-sharing mobile app, KuaiShou [7]. There are 1 million user-item interactions with 41 feature fields. **QB-Video [52]** is a multi-task dataset with true-negative behaviors released by Tencent. It contains 2.4 million interaction logs of the video recommendation on QQ-BOW. We manually add some meaningful interacted features to conduct feature selection for it. There are 12 feature fields in QB-Video we used. **AliCCP**[8] contains 52 million samples from TaoBao, and each sample has 17 feature fields. For each dataset, we randomly select 80% for training, 10% for validation, and the rest 10% for testing. Please find the detailed statistics in Table 2.

4.1.2 Evaluation Metric. We present the AUC score (Area Under the ROC Curve) and Logloss (logarithmic loss) of each task to evaluate recommendation performance. A higher AUC score and lower Logloss at 0.001-level could reveal a significant improvement, according to [13]. For n^{th} task, we denote its performance as AUC#n and Logloss#n.

4.1.3 MTRS Backbone. We choose multiple widely used MTRSs (PLE [40], MMoE [31], ESMM [32] and AITM [50]) as backbones, which is combined with our MultiSFS and other feature selection baselines. Their detailed introduction could refer to Section 5.1. We also independently train each task with FNN [54] and record recommendation results to figure out the benefit of applying MTRS, denoted as "single task".

4.1.4 Baseline. We choose three sorts of feature selection methods: (i) Traditional methods for single task, **Chi2** [24] and **LASSO** [42]; (ii) AutoML-based methods for single task, **AutoField** [47]; and (iii) Deep learning methods for MTRSs, **MSSM** [9] and **CFS** [8]. Please refer to Section 5.2 for more details on these methods.

4.1.5 Implementation Details. We implement MTRSs based on a public multi-task recommendation library[9]. All MTRSs are fairly compared with the same hyper-parameters, e.g., bottom MLP dimensions [512, 256], tower MLP dimensions [128, 64], and drop-out rate 0.2. We set the embedding dimension D as 64 for KuaiRand-Pure and QB-Video, and 32 for AliCCP. For architecture-sharing MTRSs, we keep the number of experts for each task as 8. In other words, each task has 4 shared experts in MMoE and 4 shared experts with 4 task-specific experts in PLE. For single task learning, we use the FNN with MLP layers of dimensions [512, 256, 128, 64] for each task. For our MultiSFS, the batch size is 2,048 for all steps. For feature importance calculation, we use the average score of **five** batches of data[10]. For task relation learning, we set $L = 1$ to prevent over-smoothing. All experiments are repeated 10 times with different random seeds on NVIDIA V100 Tensor Core GPU.

4.2 Overall Performance (RQ1)

In this section, we test the recommendation quality on two major tasks (with the densest positive user feedback) in each dataset, i.e., "Click" and "Like" on KuaiRand-Pure and QB-Video, as well as "CTR" and "CVR" on AliCCP. In Table 1, we demonstrate the performance improvement by adopting our MultiSFS for MTRS backbones, and in

[7]https://www.kuaishou.com/cn
[8]https://tianchi.aliyun.com/dataset/408

[9]https://github.com/easezyc/Multitask-Recommendation-Library
[10]The selection results tend to be more consistent with an increased number of batches. To mitigate the impact of outliers, we compute the average score across five batches.

Table 3: Overall performance comparison with feature selection methods.

Model	Feature Selection Method	KuaiRand-Pure				QB-Video				AliCCP			
		AUC1↑	AUC2↑	Logloss1↓	Logloss2↓	AUC1↑	AUC2↑	Logloss1↓	Logloss2↓	AUC1↑	AUC2 ↑	Logloss1↓	Logloss2↓
MMoE	Base	0.6914	0.8613	0.6416	0.0766	0.9792	0.9013	0.1527	0.0290	0.6155	0.6334	0.1597	0.2122
	Chi2	0.6925	0.8617	0.6422	0.0773	0.9793	0.8993	0.1531	0.0289	0.6113	0.6205	0.1600	**0.2073**
	LASSO	0.6916	0.8626	0.6436	0.0790	0.9793	0.8999	0.1525	0.0288	0.6113	0.6205	0.1600	**0.2073**
	AutoField	0.6873	0.8567	0.6429	0.0792	0.9790	0.8988	0.1533	0.0291	0.6143	0.6305	0.1602	0.2074
	MSSM	0.6918	0.8608	0.6450	0.0775	0.9793	0.8988	**0.1522**	0.0289	0.6169	0.6259	0.1598	0.2078
	CFS	0.6921	**0.8632**	0.6463	0.0773	0.9793	0.8994	**0.1522**	0.0288	0.6135	0.6340	0.1608	0.2133
	MultiSFS	**0.6929**	0.8628	**0.6395***	**0.0765**	0.9793	**0.9044***	0.1523	**0.0286**	**0.6182***	**0.6389***	**0.1595**	0.2099
PLE	Base	0.6916	0.8600	0.6412	0.0767	0.9792	0.8923	**0.1519**	0.0294	0.6152	0.6377	0.1595	0.2112
	Chi2	0.6920	0.8587	0.6434	0.0762	0.9791	0.9025	0.1536	0.0287	0.6136	0.6261	0.1600	0.2137
	LASSO	0.6913	0.8625	0.6446	0.0761	0.9790	0.9042	0.1535	0.0288	0.6136	0.6261	0.1600	0.2137
	AutoField	0.6866	0.8571	0.6432	0.0770	0.9788	0.8906	0.1528	0.0292	0.6133	0.6208	0.1604	0.2140
	MSSM	**0.6931**	0.8629	0.6388	0.0756	0.9790	0.8982	0.1541	0.0287	**0.6164**	0.6209	0.1593	**0.2076**
	CFS	0.6927	0.8624	**0.6398**	0.0756	0.9792	0.8998	0.1530	0.0285	0.6147	0.6350	0.1596	0.2133
	MultiSFS	0.6920	**0.8645***	0.6405	**0.0755**	**0.9794**	**0.9072***	0.1524	**0.0283**	**0.6164**	**0.6420***	**0.1592**	0.2091
ESMM	Base	0.6919	0.8598	0.6411	0.0764	0.9793	0.9048	0.1522	0.0291	0.6142	0.6292	0.1600	0.2141
	Chi2	0.6919	0.8600	0.6436	0.0764	0.9793	0.9058	**0.1515**	0.0296	0.6169	0.6267	0.1592	0.2120
	LASSO	0.6913	0.8603	0.6427	0.0783	0.9792	0.9019	0.1540	0.0293	0.6169	0.6267	0.1592	0.2120
	AutoField	0.6860	0.8561	0.6428	0.0772	0.9791	0.9021	0.1535	0.0294	0.6140	0.6256	0.1594	0.2127
	MSSM	0.6921	0.8595	0.6425	0.0763	0.9792	0.9012	0.1530	0.0295	0.6170	0.6355	0.1591	0.2105
	CFS	0.6920	**0.8633**	0.6426	**0.0760**	0.9793	0.9049	0.1516	**0.0287**	0.6172	0.6363	0.1593	**0.2097**
	MultiSFS	**0.6941***	0.8622	**0.6406**	0.0761	**0.9794**	**0.9068***	0.1516	0.0288	**0.6181**	**0.6439***	**0.1590**	0.2142
AITM	Base	0.6916	0.8589	0.6413	0.0770	0.9793	0.9008	0.1521	0.0289	0.6143	0.6325	0.1599	0.2151
	Chi2	0.6915	0.8613	**0.6402**	0.0775	0.9794	0.9014	0.1526	0.0295	0.6110	0.6208	0.1602	0.2109
	LASSO	0.6913	0.8618	0.6406	0.0772	0.9793	0.9008	0.1525	0.0298	0.6110	0.6208	0.1602	0.2109
	AutoField	0.6851	0.8479	0.6482	0.0790	0.9793	0.8994	0.1522	0.0297	0.6126	0.6237	0.1601	0.2110
	MSSM	0.6920	0.8617	0.6409	0.0771	0.9792	0.9001	0.1529	0.0297	0.6134	0.6298	0.1602	**0.2071**
	CFS	0.6923	0.8627	**0.6402**	0.0770	0.9793	0.9011	0.1528	0.0296	**0.6154**	**0.6351**	**0.1591**	0.2110
	MultiSFS	**0.6946***	**0.8659***	0.6407	**0.0761**	**0.9797**	**0.9065***	**0.1511**	**0.0288**	**0.6154**	0.6338	0.1596	0.2163

"*" indicates the statistically significant improvements over the best baseline. ↑: higher is better; ↓: lower is better.

Table 3, we compare MultiSFS with other feature selection methods. We also test the performance of "single task". We can observe that:

- As evidenced in Table 1, it can be observed that: (i) We can observe that the use of MultiSFS significantly reduces the occurrence of the seesaw phenomenon [40], particularly on QB-Video. The MTRS backbones achieve this by selectively learning relevant input features, which helps to prevent the negative transfer from redundant features and allows them to focus on exploring predictive features. (ii) MultiSFS has been proven to enhance various MTRS backbones, and it has been observed to be particularly compatible with output-sharing models (ESMM and AITM), where it consistently exhibits a marked improvement. Independent bottom networks of these models, which learn task-specific features, enhancing feature selection benefits.
- The results presented in Table 3 demonstrate that MultiSFS outperforms other feature selection methods in most experimental settings. In fact, the best-performing baselines could not surpass MultiSFS with statistical significance. To provide a more nuanced explanation, (i) Chi2 and LASSO perform feature selection independently for each task, which only results in a modest improvement due to the lack of consideration for task interdependence. (ii) AutoField's performance is hindered by its practice of dropping fields with numerous feature values, which are typically indicative, thereby resulting in negative transfer effects between features [8, 9]. This leads to misguided bi-level optimization of AutoField. (iii) MSSM and CFS achieve similar results due to their similar feature selection layer design. However, CFS outperforms MSSM with the addition of causal inference guidance. (iv) MultiSFS performs better than CFS and MSSM since we select feature fields prior to train MTRS models, totally avoiding the negative effects of suboptimal feature fields. Furthermore, MultiSFS takes into account task relationships, which further enhances its performance.

4.3 Generalization Ability Analysis (RQ2)

To investigate the generalization ability of MultiSFS in terms of the number and type of tasks, we conduct experiments with different task settings on KuaiRand-Pure. In particular, we select the following two representative task settings:

- **"Click" and "Hate"**. In Section 4.2, we investigate the feature selection effect of MultiSFS with two positively related tasks. For example, both "Click" and "Like" show the user's appreciation of a recommended short video in KuaiRand-Pure. Thus, we would also study MultiSFS in other settings, such as negatively related tasks "Click" and "Hate".
- **"Click", "Like", "Comment"**. We also would like to investigate the model performance when there are more than two recommendation tasks. Thus, we select the three major tasks (i.e., top3 tasks with the most positive feedback from users) in KuaiRand-Pure.

We compare our MultiSFS with the best-performed baseline (CFS [8]) based on MTRS backbones. The results are presented in Table 4. We can make the following observations:

- MultiSFS has demonstrated a significant improvement in negatively related tasks. Our findings reveal that architecture-sharing

Table 4: Generalization ability test with different task settings on KuaiRand-pure.

Task Settings	Click & Hate				Click + Like + Comment					
Metrics	AUC1↑	AUC2 ↑	Logloss1↓	Logloss2↓	AUC1↑	AUC2 ↑	AUC3↑	Logloss1↓	Logloss2↓	Logloss3 ↓
Single Task	0.6935	0.7554	0.6413	**0.0049**	0.6923	0.8592	0.7354	0.6397	0.0761	0.0170
MMoE	0.6821	0.7485	0.6637	0.0053	0.6903	0.8563	0.7247	0.6389	0.0784	0.0173
MMoE + CFS	0.6864	0.7164	0.6610	0.0051	0.6913	0.8605	0.7245	0.6436	0.0772	0.0171
MMoE + MultiSFS	0.6868	0.7533	0.6605	0.0052	0.6907	0.8613	0.7354	0.6381	0.0783	0.0172
PLE	0.6819	0.7342	0.6706	0.0051	0.6905	0.8605	0.7473	0.6421	0.0768	0.0168
PLE + CFS	0.6694	0.7351	0.7120	0.0057	0.6924	0.8544	0.7369	0.6417	0.0780	0.0171
PLE + MultiSFS	0.6842	0.7491	0.6646	0.0053	0.6915	0.8602	0.7494	0.6401	0.0770	0.0173
ESMM	0.6823	0.7568	0.6635	0.0055	0.6910	0.8556	0.7559	0.6393	0.0770	0.0173
ESMM + CFS	0.6922	**0.7678**	0.6434	0.0051	0.6920	0.8547	0.7576	0.6413	0.0787	0.0178
ESMM + MultiSFS	**0.6939**	0.7651	**0.6387***	0.0052	**0.6946***	0.8559	0.7563	**0.6374**	0.0778	0.0186
AITM	0.6775	0.7343	0.6756	0.0053	0.6911	0.8575	0.7423	0.6413	0.0771	0.0169
AITM + CFS	0.6823	0.7366	0.6384	0.0050	0.6924	0.8610	0.7522	0.6421	0.0766	0.0170
AITM + MultiSFS	0.6911	0.7563	0.6456	0.0051	0.6939	**0.8645***	**0.7577**	0.6392	**0.0757**	**0.0169**

"*" indicates the best performance with statistically significant improvements. ↑: higher is better; ↓: lower is better.

Table 5: Efficiency analysis on KuaiRand-Pure.

	MSSM (s)	CFS (s)	Ours-FS (s)	Ours-GNN (s)
MMoE	73.38	75.79	1.03	35.505
PLE	93.09	105.31	1.25	
ESMM	42.92	44.05	1.22	
AITM	46.48	51.58	1.51	

models (MMoE and PLE) perform less effectively than output-sharing models (ESMM and AITM), while "single task" outperforms all MTRS models. This can be attributed to the severity of the negative transfer problem in negatively related tasks, which is aggravated by the shared bottom parameters in MMoE and PLE. The MultiSFS effectively mitigates negative transfer from the feature input layer, resulting in a remarkable improvement by reducing shared features for negatively related tasks.

- With three tasks, the seesaw phenomenon in architecture-sharing models is more significant. Our findings suggest that output-sharing models focus mainly on the last task, with previous tasks serving as auxiliary tasks. The application of feature selection methods alleviates this phenomenon by reducing shared features and mitigating negative transfer across multiple tasks. Additionally, incorporating task-specific features enhances performance, particularly for the first few tasks in output-sharing models.

4.4 Efficiency Analysis (RQ3)

This section compares the model efficiency. In Table 5, we present the feature selection time for different MTRS backbones based on "Click + Like" of KuaiRand-Pure. For compared methods, we record the time to generate feature selection results, which is also the training time for the model to converge. For MultiSFS, we record the one-shot feature selection and model construction time ("Ours-FS") as well as field-wise task relation learning time ("Ours-GNN"). The results show that MultiSFS's single-shot feature selection requires little time, while task relation learning takes longer, but remains more efficient than baselines as it is a simple GNN independent of MTRS backbones. In addition, CFS requires more time than MSSM due to the use of causal inference in every step.

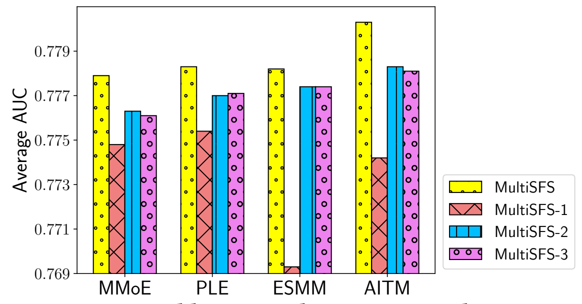

Figure 2: Ablation study on KuaiRand-Pure.

4.5 Ablation Study (RQ4)

We aim to figure out crucial components of MultiSFS in this section. We design three variants of MultiSFS:

- **MultiSFS-1:** To validate our assertion in Section 3.2 that the training process unequally explores the value of features and leads to suboptimal feature selection, we evaluate the feature importance after training MTRS to converge.
- **MultiSFS-2 & MultiSFS-3:** To explore the impact of field-wise task relations, we introduce these two variants. MultiSFS-2 uses Pearson correlations between task labels as the task relation (a unified value for all fields), while MultiSFS-3 directly selects features with s_{tn} without task relations.

We test the performance on "Click + Like" tasks of KuaiRand-Pure, and the results are visualized in Figure 2. We can observe that:

- The performance of MultiSFS-1 significantly drops. This variant drops some predictive feature fields since pre-training misleads the selection, as we claim in Section 3.2.
- MultiSFS-2 and MultiSFS-3 almost perform the same. The limited availability of positive samples results in a Pearson correlation that is insufficient to have a significant impact on feature importance. Additionally, the observed decline in the performance of both models serves as evidence for the efficacy of incorporating field-level task relations in our analysis.

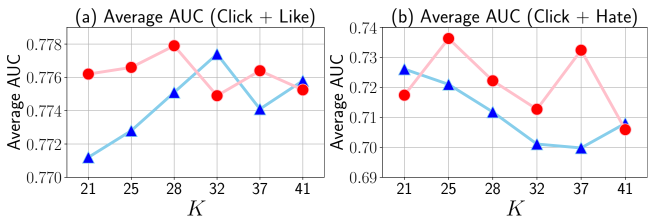

Figure 3: Parameter analysis on KuaiRand-Pure.

4.6 Hyper-parameter Study (RQ5)

In this section, we tend to investigate the influence of our model's unique hyper-parameter $\{k_t\}$, i.e., the number of selected feature fields for tasks. In this experiment, we set $k_1 = \cdots = k_T = K$ and choose K from $0.5N, 0.6N, \ldots, N$, with N the total number of feature fields. In other words, when $K = N$, we do not conduct feature selection. We fix all other hyper-parameters and select the optimal feature fields via MultiSFS on KuaiRand-Pure, where $N = 41$. We test the performance with task settings "Click + Like" and "Click + Hate". The results are visualized in Figure 3, where the X-axis is K and the Y-axis is the average AUC of two tasks. The red line is the result of applying MultiSFS to AITM, and the blue one is that of PLE. We can find that:

- The optimal value of the parameter K^* varies depending on the MTRSs and the specific task setting. However, it is noted that the optimal value of K^* with regard to the average AUC metric is similar for all MTRSs when examining a specific task setting.
- It is observed that when $K > K^*$, the presence of redundant feature fields negatively impacts the performance of the model. Conversely, when $K < K^*$, some of the predictive feature fields are removed, causing a performance decline. However, note that visual representations of this trend are not always strictly evident since the optimal value of K^* may not be applicable to all tasks.

5 RELATED WORK

We briefly introduce related works in this section from views of multi-task recommendation models and feature selection methods.

5.1 Multi-task Recommendation

Architecture-sharing models usually share the information across tasks by sharing a part of bottom parameters. The most straight forward idea is sharing all bottom parameters, known as shared-bottom [7]. Following works split the whole bottom network into sub-networks and focus on devising different parameter-sharing strategies across sub-networks, including sub-network routing [9, 30, 33, 37] or pruning [3, 39], and adopting gate mechanisms [31, 40, 66]. For example, MMoE [31] uses several gates to control the usage of sub-networks (experts). Based on MMoE, PLE [40] explicitly divides experts into groups of 'task-specific' and 'shared'.

Output-sharing models are mainly designed for cascading tasks. They only transfer the common information from former tasks to the next [32, 43, 46, 48, 50]. In our paper, we select ESMM [32] and AITM [50] for comparison. ESMM [32] shares the 'click' information to post-click conversion rate (CVR) prediction by element-wise product to tackle the sample selection bias problem in CVR estimation. AITM [50] designs an adaptive information transfer module to learn which part of the information to transfer for different conversion stages in longer user behavior sequences.

However, all the above models neglect to distinguish different contributions of feature fields to taskswhich may lead to the negative transfer problem [8, 9]. Our MultiSFS is a model-agnostic feature selection framework suit for above mentioned MTRSs.

5.2 Feature Selection

Feature selection aims to select predictive feature fields for constructing prediction models. Traditional feature selection methods include the filter, the embedded, and the wrapper [14]. Filters usually screen out predictive feature fields through specific criterion, such as Chi2 score [24]. For embedded methods, the selection process is conducted as the prediction model works. LASSO [42] and gradient-boosting methods [5] are famous embedded feature selection frameworks. Wrappers evaluate feature set candidates by black-box models, which are often evolutionary algorithms [12]. Inspired by the success of reinforcement learning (RL) [57, 62–64] and automated machine learning (AutoML) techniques in recommendations [17, 21, 23, 25, 38, 56, 58–60, 65], RL-based [10, 26, 27, 61] and AutoML-based methods [22, 47] are applied for feature selection. Separately applying these methods in MTRS ignores task relations.

Several feature extraction [41, 51] or feature selection [8, 9, 20, 34] methods are designed for MTRSs. For example, MSSM [9] selects features by generating sparse masks through ReLU(Tanh($\cdot$)). Based on a similar operation Softmax(ReLU(Tanh($\cdot$))), CFS [8] further introduce causal inference techniques to guide the feature selection. However, both of the above methods introduce extra model parameters and optimization steps. Our MultiSFS is the first to explicitly model field-level task relations and efficiently conduct feature selection in a single-shot manner.

6 CONCLUSION

In this paper, we propose MultiSFS, a novel feature selection framework for multi-task recommendations. MultiSFS first scores all feature fields for each task with a single forward-backward pass. Then, MultiSFS learns field-wise task relations by constructing a data-task bipartite graph and learning the field-level task representations. Finally, MultiSFS selects predictive features based on single-shot scores and task relations. Extensive experiments validate the effectiveness of MultiSFS, as evidenced by its performance on multiple real-world datasets under different task settings. In addition, MultiSFS presents outstanding efficiency compared with traditional feature selection methods. These findings highlight the usefulness of MultiSFS for multi-task recommendations, offering a comprehensive and effective solution for feature selection.

ACKNOWLEDGEMENT

This research was supported by Huawei (Huawei Innovation Research Program). We thank MindSpore [1] for the partial support of this work, which is a new deep learning computing framework. It was also partially supported by APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for New Faculty of City University of Hong Kong), SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074), CityU - HKIDS Early Career Research Grant (No.9360163).

REFERENCES

[1] 2020. MindSpore. https://www.mindspore.cn/
[2] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. 2021. Zero-cost proxies for lightweight nas. *International Conference on Representation Learning (ICLR)* (2021).
[3] Ting Bai, Yudong Xiao, Bin Wu, Guojun Yang, Hongyong Yu, and Jian-Yun Nie. 2022. A Contrastive Sharing Model for Multi-Task Recommendation. In *Proceedings of the ACM Web Conference 2022.*
[4] Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303* (2017).
[5] Leo Breiman. 1997. *Arcing the edge.* Technical Report.
[6] Kaidi Cao, Jiaxuan You, and Jure Leskovec. 2022. Relational multi-task learning: Modeling relations between data and tasks. In *International Conference on Representation Learning (ICLR).*
[7] Rich Caruana. 1997. Multitask learning. *Machine learning* (1997).
[8] Zhongde Chen, Ruize Wu, Cong Jiang, Honghui Li, Xin Dong, Can Long, Yong He, Lei Cheng, and Linjian Mo. 2022. CFS-MTL: A Causal Feature Selection Mechanism for Multi-task Learning via Pseudo-intervention. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management.* 3883–3887.
[9] Ke Ding, Xin Dong, Yong He, Lei Cheng, Chilin Fu, Zhaoxin Huan, Hai Li, Tan Yan, Liang Zhang, Xiaolu Zhang, et al. 2021. MSSM: a multiple-level sparse sharing model for efficient multi-task learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2237–2241.
[10] Wei Fan, Kunpeng Liu, Hao Liu, Pengyang Wang, Yong Ge, and Yanjie Fu. 2020. Autofs: Automated feature selection via diversity-aware interactive reinforcement learning. In *2020 IEEE International Conference on Data Mining (ICDM).* IEEE, 1008–1013.
[11] Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. 2022. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management.* 3953–3957.
[12] Pablo M Granitto, Cesare Furlanello, Franco Biasioli, and Flavia Gasperi. 2006. Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products. *Chemometrics and intelligent laboratory systems* (2006).
[13] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence.* 1725–1731.
[14] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* (2003).
[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).
[16] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval.* 355–364.
[17] Wei Jin, Xiaorui Liu, Xiangyu Zhao, Yao Ma, Neil Shah, and Jiliang Tang. 2022. Automated Self-Supervised Learning for Graphs. In *10th International Conference on Learning Representations (ICLR 2022).*
[18] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning.* PMLR, 1885–1894.
[19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2019. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations.* https://openreview.net/forum?id=B1VZqjAcYX
[20] Lingjie Li, Manlin Xuan, Qiuzhen Lin, Min Jiang, Zhong Ming, and Kay Chen Tan. 2022. An Evolutionary Multitasking Algorithm with Multiple Filtering for High-Dimensional Feature Selection. *arXiv preprint arXiv:2212.08854* (2022).
[21] Muyang Li, Zijian Zhang, Xiangyu Zhao, Wanyu Wang, Minghao Zhao, Runze Wu, and Ruocheng Guo. 2023. AutoMLP: Automated MLP for Sequential Recommendations. In *Proceedings of the Web Conference 2023.*
[22] Weilin Lin, Xiangyu Zhao, Yejing Wang, Tong Xu, and Xian Wu. 2022. AdaFS: Adaptive Feature Selection in Deep Recommender System. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 3309–3317.
[23] Weilin Lin, Xiangyu Zhao, Yejing Wang, Yuanshao Zhu, and Wanyu Wang. 2023. AutoDenoise: Automatic Data Instance Denoising for Recommendations. In *Proceedings of the Web Conference 2023.*
[24] Huan Liu and Rudy Setiono. 1995. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of 7th IEEE international conference on tools with artificial intelligence.* IEEE, 388–391.
[25] Haochen Liu, Xiangyu Zhao, Chong Wang, Xiaobing Liu, and Jiliang Tang. 2020. Automated Embedding Size Search in Deep Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2307–2316.
[26] Kunpeng Liu, Yanjie Fu, Pengfei Wang, Le Wu, Rui Bo, and Xiaolin Li. 2019. Automating feature subspace exploration via multi-agent reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 207–215.
[27] Kunpeng Liu, Yanjie Fu, Le Wu, Xiaolin Li, Charu Aggarwal, and Hui Xiong. 2021. Automated feature selection: A reinforcement learning perspective. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[28] Qi Liu, Qian Xu, Vincent W Zheng, Hong Xue, Zhiwei Cao, and Qiang Yang. 2010. Multi-task learning for cross-platform siRNA efficacy prediction: an in-silico study. *BMC bioinformatics* (2010).
[29] Ziru Liu, Jiejie Tian, Qingpeng Cai, Xiangyu Zhao, Jingtong Gao, Shuchang Liu, Dayou Chen, Tonghao He, Dong Zheng, Peng Jiang, et al. 2023. Multi-Task Recommendations with Reinforcement Learning. In *Proceedings of the Web Conference 2023.*
[30] Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H Chi. 2019. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 216–223.
[31] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining.* 1930–1939.
[32] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* 1137–1140.
[33] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 3994–4003.
[34] Alejandro Newell, Lu Jiang, Chong Wang, Li-Jia Li, and Jia Deng. 2019. Feature partitioning for efficient multi-task architectures. *arXiv preprint arXiv:1908.04339* (2019).
[35] Liang Qu, Yonghong Ye, Ningzhi Tang, Lixin Zhang, Yuhui Shi, and Hongzhi Yin. 2022. Single-shot Embedding Dimension Search in Recommender System. *Proceedings of the 45th International ACM SIGIR conference on research and development in Information Retrieval* (2022).
[36] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th international conference on data mining (ICDM).* IEEE, 1149–1154.
[37] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142* (2017).
[38] Fengyi Song, Bo Chen, Xiangyu Zhao, Huifeng Guo, and Ruiming Tang. 2022. AutoAssign: Automatic Shared Embedding Assignment in Streaming Recommendation. In *IEEE International Conference on Data Mining (ICDM).* IEEE, 458–467.
[39] Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. Learning sparse sharing architectures for multiple tasks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 8936–8943.
[40] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems.* 269–278.
[41] Xuewen Tao, Mingming Ha, Xiaobo Guo, Qiongxu Ma, Hongwei Cheng, and Wenfang Lin. 2023. Task Aware Feature Extraction Framework for Sequential Dependence Multi-Task Learning. *arXiv preprint arXiv:2301.02494* (2023).
[42] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* (1996).
[43] Hao Wang, Tai-Wei Chang, Tianqiao Liu, Jianmin Huang, Zhichao Chen, Chao Yu, Ruopeng Li, and Wei Chu. 2022. ESCM2: Entire Space Counterfactual Multi-Task Model for Post-Click Conversion Rate Estimation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 363–372.
[44] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17.* 1–7.
[45] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval.* 1001–1010.
[46] Yanshi Wang, Jie Zhang, Qing Da, and Anxiang Zeng. 2020. Delayed feedback modeling for the entire space conversion rate prediction. *arXiv preprint arXiv:2011.11826* (2020).
[47] Yejing Wang, Xiangyu Zhao, Tong Xu, and Xian Wu. 2022. Autofield: Automating feature selection in deep recommender systems. In *Proceedings of the ACM Web Conference 2022.*

[48] Hong Wen, Jing Zhang, Yuan Wang, Fuyu Lv, Wentian Bao, Quan Lin, and Keping Yang. 2020. Entire space multi-task modeling via post-click behavior decomposition for conversion rate prediction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 2377–2386.

[49] Haolun Wu, Yingxue Zhang, Chen Ma, Wei Guo, Ruiming Tang, Xue Liu, and Mark Coates. 2022. Intent-aware Multi-source Contrastive Alignment for Tag-enhanced Recommendation. *arXiv preprint arXiv:2211.06370* (2022).

[50] Dongbo Xi, Zhen Chen, Peng Yan, Yinger Zhang, Yongchun Zhu, Fuzhen Zhuang, and Yu Chen. 2021. Modeling the sequential dependence among audience multi-step conversions with multi-task learning in targeted display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3745–3755.

[51] Shen Xin, Yuhang Jiao, Cheng Long, Yuguang Wang, Xiaowei Wang, Sen Yang, Ji Liu, and Jie Zhang. 2022. Prototype Feature Extraction for Multi-task Learning. In *Proceedings of the ACM Web Conference 2022*.

[52] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun YU, Bo Hu, Zang Li, Yu Xu, and Xiaohu Qie. 2022. Tenrec: A Large-scale Multipurpose Benchmark Dataset for Recommender Systems. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=PfuW84q25y9

[53] Xiao-Tong Yuan, Xiaobai Liu, and Shuicheng Yan. 2012. Visual classification with multitask joint sparse representation. *IEEE Transactions on Image Processing* (2012).

[54] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data: –A Case Study on User Response Prediction. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*. Springer, 45–57.

[55] Yu Zhang and Qiang Yang. 2017. Learning sparse task relations in multi-task learning. In *Proceeding of the AAAI Conference on Artificial Intelligence*.

[56] Xiangyu Zhao. 2022. Adaptive and automated deep recommender systems. *ACM SIGWEB Newsletter* Spring (2022), 1–4.

[57] Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Hui Liu, and Jiliang Tang. 2021. DEAR: Deep Reinforcement Learning for Online Advertising Impression in Recommender Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 750–758.

[58] Xiangyu Zhao, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, and Chong Wang. 2021. AutoLoss: Automated Loss Function Search in Recommendations. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3959–3967.

[59] Xiangyu Zhao, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Xiwang Yang. 2021. Autoemb: Automated embedding dimensionality search in streaming recommendations. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 896–905.

[60] Xiangyu Zhao, Haochen Liu, Hui Liu, Jiliang Tang, Weiwei Guo, Jun Shi, Sida Wang, Huiji Gao, and Bo Long. 2021. AutoDim: Field-aware Embedding Dimension Searchin Recommender Systems. In *Proceedings of the Web Conference 2021*. 3015–3022.

[61] Xiaosa Zhao, Kunpeng Liu, Wei Fan, Lu Jiang, Xiaowei Zhao, Minghao Yin, and Yanjie Fu. 2020. Simplifying reinforced feature selection via restructured choice strategy of single agent. In *Proceeding of the IEEE International Conference on Data Mining*.

[62] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. In *Proceedings of the 12th ACM Recommender Systems Conference*. ACM, 95–103.

[63] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1040–1048.

[64] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2017. Deep Reinforcement Learning for List-wise Recommendations. *arXiv preprint arXiv:1801.00209* (2017).

[65] Chenxu Zhu, Bo Chen, Huifeng Guo, Hang Xu, Xiangyang Li, Xiangyu Zhao, Weinan Zhang, Yong Yu, and Ruiming Tang. 2023. AutoGen: An Automated Dynamic Model Generation Framework for Recommender System. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 598–606.

[66] Yongchun Zhu, Yudan Liu, Ruobing Xie, Fuzhen Zhuang, Xiaobo Hao, Kaikai Ge, Xu Zhang, Leyu Lin, and Juan Cao. 2021. Learning to expand audience via meta hybrid experts and critics for recommendation and advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4005–4013.